Bayreuther Arbeitspapiere zur Wirtschaftsinformatik

Björn Schnizler / Dirk Neumann / Daniel Veit / Michael Reinicke / Werner Streitberger / Torsten Eymann / Felix Freitag / Isaac Chao / Pablo Chacin

# A Theoretical and Computational Basis for CATNETS

Bayreuth Reports on Information Systems Management

UNIVERSITÄT BAYREUTH

Die Arbeitspapiere des Lehrstuhls für Wirtschaftsinformatik dienen der Darstellung vorläufiger Ergebnisse, die i. d. R. noch für spätere Veröffentlichungen überarbeitet werden. Die Autoren sind deshalb für kritische Hinweise dankbar**.**

The Bayreuth Reports on Information Systems Management comprise preliminary results which will usually be revised for subsequent publications. Critical comments would be appreciated by the authors.

**Authors:**

Björn Schnizler (University of Karlsruhe)
Dirk Neumann (University of Karlsruhe)
Daniel Veit (University of Karlsruhe)
Michael Reinicke (University of Bayreuth)
Werner Streitberger (University of Bayreuth)
Torsten Eymann (University of Bayreuth)
Felix Freitag (Universitat Politecnica de Catalunya)
Isaac Chao (Universitat Politecnica de Catalunya)
Pablo Chacin (Universitat Politecnica de Catalunya)

**Managing Assistant and Contact:**

Raimund Matros
Universität Bayreuth
Lehrstuhl für Wirtschaftsinformatik (BWL VII)
Prof. Dr. Torsten Eymann
Universitätsstrasse 30
95447 Bayreuth
Germany

Email: raimund.matros@uni-bayreuth.de

# Contents

# Chapter 1

# Introduction

What makes Economics so attractive for computing environments is that its central research question lies in the effective allocation of resources, provided by suppliers and in demand by customers. In computing environments like Grid Computing, the resources in question are processor time or storage space, while the economic actors are computers or web services [BAG01]. It appears that, by just implementing markets in computing environments, the satisfying ability of economics might be viable for creating cost-effective computer architectures. However, between the mostly descriptive economic concept and the normative technical implementation lies a fundamental gap, requiring selective choice of how actors, resources, goods, and markets are modeled and embedded in a technical environment.

There are several competing descriptive approaches to how economic resource allocation mechanisms work. In general, Economics is essentially all about the coordination of systems consisting of utility-maximizing agents, who satisfy their needs using some mechanism for solving a distributed resource allocation problem. The effect of this mechanism is a state where prices are set, so that supply and demand is perfectly balanced, and the number of transactions is maximized [KSBE00]. All implementation attempts try either to recreate the mechanism, or to achieve the effect by using another mechanism, adding some side condition like zero communication costs or a steady environment state. Adam Smith's proverbial invisible hand [Smi79] was a first concept of a decentralized mechanism without a coordinator, but Smith gave no implementation of that mechanism. A century later, Leon Walras [Wal54] introduced a central auctioneer, who iteratively solved the allocation problem out of total knowledge of supply and demand. With this mechanism, Walras was able to generate the desired equilibrium effect. Most of today's economic research relies on Walras' tatonnement process as a valid picture of the mechanism, which influences also the possible realization in computing environments. An example is the realization by Wellman [Wel93], titled Market-Oriented Programming (MOP), in an distributed artificial intelligence (DAI) environment. Wellman takes the notion that "an economy is a multi agent system" literally; the distributed agents individually

compute their utility functions and post that information to a centralized Walrasian auctioneer. During the computation process, interrelated markets are successively brought to near-equilibrium by the auctioneer, with the final general equilibrium effect as the "gold standard" to achieve. MOP has been successfully used in electricity markets [Ygg98], for multi-commodity flow problems [Wel93], supply chain management [WW03] or for negotiations about the quality of service in multimedia networks [YUWI95].

In contrast, Economics research on self-organization still aims at explaining the mechanism of the invisible hand, e.g. Agent-based Computational Economics [Tes02]. Actually, there is growing interest in using self-organization, as indicated by the start of large industrial research concepts like IBM's Autonomic Computing initiative. Autonomic Computing uses a biological paradigm as a design and control metaphor, the autonomic nervous system [KC03]. If the mechanisms underlying Hayek's spontaneous order concept [HBKC89] can be properly understood, it might be possible to build large Autonomic information systems using the Catallaxy approach, where artificial entities coordinate themselves, just as human economy participants do in the real world. For a start, we have to discuss whether the desired effects of Autonomic Computing are achievable (and describable) using economic terms.

IBM's Autonomic Computing Manifesto (IBM) describes seven characteristics, which self-adapting systems should exhibit. The core characteristics are contained in the so-called CHOP cycle of self-configuring, self-healing, self-optimizing and self-protection capabilities. The self-configuration property is indicated in the variation of prices when adding or removing service providers (cf. the different density regimes). The self-healing of the system is apparent in case a service provider instance shuts down or a network connection gets broken (cf. the different dynamics regimes). The application is self-optimizing, in that the agents constantly attempt to change their strategies towards the maximum utility-eliciting negotiation positions, which respectively lay on the total supply and demand curves. The self-protection of the application finally can be reached by including security mechanisms like reputation tracking [EPS00]), which are effective in separating malicious and under performing agents. In addition, viewing Autonomic Computing systems as Economic systems has some merits, too. The main applications for AC systems will be deeply rooted in a business context. With a biological background, you need to find biological translations for conceptual data structures and functionality for describing success, utility, or business goals. This is not a trivial process, and may lead to semantic loss underway.

For example, the business goal of maintaining availability (to prevent loss of profit in the case of server downtime), may be translated biologically as "staying alive". However, the semantics of both differ – deliberately shutting down a biological AC system may qualify for murder, while in economic terms, shutting down a system means buying it out of business – with the programmer defining what the currency is. The key to this semantic shift is to view the "market" as an emergent mechanism of coordinating and matching supply and demand offers, and market participants as rationally-bounded, self-interested individuals, like in Neo-Austrian [EPS00] or Neo-Institutional Economics

[NC90][FR98]. As we move on to technology, which allows us to map unstructured semantic knowledge in large and very dynamic systems, we have to look for decentralized self-organization, not at least for reasons of exponentially increasing "costs of ownership" [TRB99]. Given a highly complex and dynamic ALN infrastructure, scalability and the management of a great number of heterogeneous resources are supposed to be challenging issues of future ALNs and computing systems in general. Ubiquitous computing [Wei91] envisions trillions of computing devices connecting and interacting with each other; Grid Computing [SP03]) envisions millions of networked processors. To handle the complexity and scale of such systems, the necessity of a centralized management could easily turn the vision into a "nightmare" [KC03]. The solution is not necessarily a question of overcoming semantic gaps or problems of multi-attribute optimization. Discovering and selecting web services from huge numbers of unreliable candidates alone is challenging enough.

## 1.1   Motivation

The increasing interconnection between computers through the Internet has emerged the vision of Application Layer Networks (ALN). Application Layer Networks comprise an abstract view on overlay networks (e.g. Peer-to-Peer networks, Grid infrastructures) on top of the TCP/IP protocol. Their common characteristic is the redundant, distributed provisioning and access of data, computation or application services, while hiding the heterogeneity of the service network from the user's view [ERA+03].

Promising examples for Application Layer Networks are Computational Grids. In the Computational Grid, computer resources such as processors or hard disks can be accessed in analogy to the power grid in a plug-and-play environment. A user has access to a reliable virtual computer, which consists of many heterogeneous computer resources. These resources are not visible to the user - such as a consumer of electric power is unaware of how the demanded electricity is being generated and thereafter transmitted to the power socket.

At the moment, most of the research done in the area of Application Layer Networks focuses in particular on the hardware and software infrastructure, such that from a technical point of view, "the access to resources is dependable, consistent, pervasive, and inexpensive" [FK04]. Nonetheless, there are still barriers preventing the deployment of large-scaled Peer-to-Peer networks or Computational Grids.

One of the key issues in building such networks is to determine which computer resources are allocated to which service. Most existing approaches such as Legion [COBW00], Condor [FTF+01], or Gnutella [FFRS02, KSTT04] employ optimization algorithms, which allocate resources based on static system specific cost functions [BAGS02, BAV04].

However, static system specific cost functions typically lead to economically ineffi-

cient allocations [BAGS02, BAV04]. These functions do not guarantee that those demanding users will receive their supplied resources who value them highest. Furthermore, these functions ignore the fact that users owning resources only have incentives offering resources, if they are adequately compensated. Compensation requires determining how the supplied resources are allocated among potential buyers and how the prices for the resources are set. However, for implementing economic efficient Application Layer Networks both described aspects are crucial.

Recently, the application of market mechanisms for allocating services and resources in Application Layer Networks has been increasingly suggested [BAGS02, WPBB01a]. According to Hurwicz, markets can be an efficient institution to allocate resources (Pareto-) optimal [Hur72]. This is achieved by the interplay of demand and supply and due to the information feedback inherent to the price system. As such, the application of market mechanisms to Application Layer Networks as an allocation and scheduling mechanism is deemed promising.

## 1.2 CATNETS Scenario

In Application Layer Networks, participants offer and request application services and computing resources of different complexity and value - creating interdependent markets. In CATNETS, these complex interdependencies are broken down into two types of interrelated markets:

- (1) a resource market - which involves trading of computational and data resources, such as processors, memory, etc, and

- (2) a service market - which involves trading of application services.

This distinction between resource and service is necessary to allow different instances of the same service to be hosted on different resources. It also enables a given service to be priced based on the particular resource capabilities that are being made available by some hosting environment.

The scenario that is envisioned in CATNETS is, that there is a set of basic services (e.g. services to create a PDF or to convert a MP3 file), a set of complex services demanding these services for a specific job (e.g. an application wants to create a PDF file), and a set of resource services capable providing computational resources for executing these services (e.g. a processor, main memory, and a hard disk for creating the PDF file). However, an agent that is requesting a service is unaware of the resources the requested service requires to be carried out.

Figure 1.1 illustrates the CATNETS scenario. A complex service is requesting a PDF creator service, which will be allocated to the agent. Furthermore, the required resources
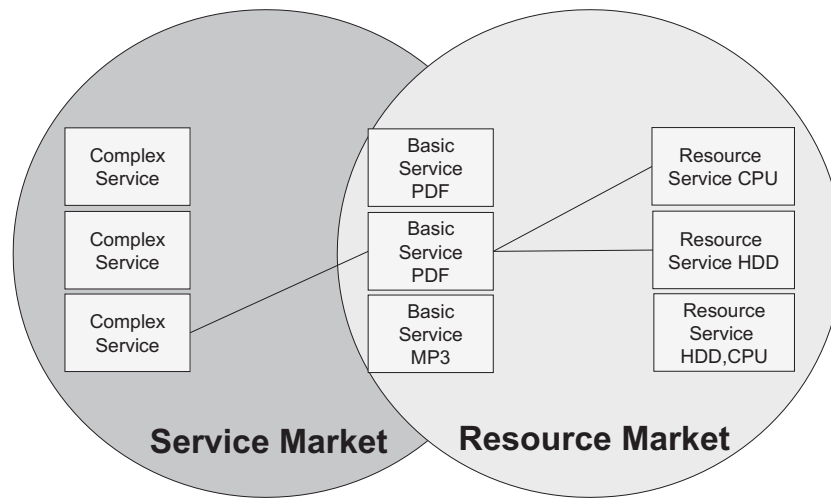
Figure 1.1: CATNETS Scenario: Service Market and Resource Market

(CPU and hard disk) are allocated to the service. The service acts as a trading intermediary, i.e. the service knows what the agents are demanding and which resources are available for executing the services.

In the next section the CATNETS approach is compared with other resource management systems.

## 1.3 State of the art: Market-based Resource Management Systems

There are several ways to classify market-based resource management systems. In the following the classification will follow the lines of Buyya et al and will sketch the related projects to CATNETS [BSGA01] [YB04]. Essentially, the taxonomy is structured around the market mechanisms, which are incorporated in the resource management system. It will be differentiated into the following market mechanisms:

**Posted Price** In posted price settings the price is revealed openly to all participants. As we have shown in our preceding motivation, the posted price is inadequate from an economic point of view if demand and/or supply are strongly fluctuating.

**Commodity Market** In a commodity market, the resource owners determine their pricing policy and the user determines accordingly his/her amount of resources to consume. Pricing can depend on the parameters such as usage time (e.g. peak-load pricing) or usage quantity (e.g. price discrimination) [WPBB01b]. In many cases, it is referred to flat rates, which boils down to fixed price until a certain amount of resources or a certain time is reached.

**Bargaining** In bargaining markets, resource owners and users negotiate bilaterally for mutually agreeable prices [SS03] [Wol88] [FS01]. By gradually lowering their claims, resource owners and users eventually reach an agreement. Logrolling for several attributes can also be included in the bargaining protocol [KSL04] [RZ94].

**Contract Net Protocol** In the contract net protocol the user advertises its demand and invites resource owners to submit bids. Resource owners check these advertisements with respect to their requirements. In case, the advertisement is favorable the resource owners respond with bids. The user consolidates all bids, compares them and selects the most favorable bids(s).

**Proportional Share** Proportional Share assigns resources proportionally to the bids submitted by the users.

**Auctions** Auctions are mediated market mechanisms. An auctioneer collects bids from either one market side (buyers or sellers) or from both. According to the auction rules (which are known to all bidders) the auctioneer allocates the resources. Typical one-sided auctions are the English, First-Price Sealed Bid, Dutch and Vickrey auction. Two sided auctions are the Double Auctions.

A survey using the above mentioned taxonomy over selected market-based resource management systems are summarized in the following [Neu05].

| Computing Platform | Market-based RMS | Market Mechanism | Description |
|---|---|---|---|
| Parallel and Distributed Systems | SPAWN | Auction | The SPAWN system provides a market mechanism for trading CPU times in a network of workstations [WHH$^+$92]. SPAWN treats computer resources as standardized commodities and implements a standard Vickrey auction. It is known from auction theory that the Vickrey auction attains (1) truthful preference revelation and (2) an efficient allocation of resources [Kri02]. However, SPAWN does not make use of the generalized Vickrey auction, which can cope with complementarities. Furthermore, the Vickrey auction can traditionally neither cope with multiple attributes nor with different time slots. |

| Peer-to-Peer | Stanford Peers | Bargaining | The Stanford Peers model is a Peer-to-Peer system which implements auctions within a cooperative bartering model in a cooperative sharing environment [WBPB03]. It simulates storage trading for content replication and archiving. It demonstrates distributed resource trading policies based on auctions by simulation. |
|---|---|---|---|
| Internet | POPCORN | Auction | POPCORN provides an infrastructure for global distributed computation [RN98] [NLRC88]. POPCORN mainly consists of three entities: (i) A parallel program which requires CPU time (buyer), (ii) a CPU seller, and (iii) a market which serves as meeting place and matchmaker for the buyers and sellers. Buyers of CPU time can bid for one single commodity, which can be traded executing a Vickrey auction repeatedly. POPCORN obviously suffers under the same shortcomings as SPAWN. |
| Grids | Bellagio | Auction | Bellagio is intended to serve as a resource discovery and resource allocation system for distributed computing infrastructures. Users express preferences for resources using a bidding language, which support XOR bids. The bids are formulated in virtual currency. The auction employed in Bellagio is periodic. Bids from users are only accepted as long as enough virtual currency is left [ACSV04]. |
| Grids | G-Commerce | Commodity market, auction | G-Commerce provides a framework for trading computer resources (CPU and hard disk) in commodity markets and Vickrey auctions [WPBB01a] [WPBB01b] [WBPB03]. While the Vickrey auction has the aforementioned shortcomings in grid, the commodity market typically works with standardized products. Additionally, the commodity market cannot account for the complementarities among the resources, as only one leg of the bundle is auctioned off, exposing the bidder to the threshold risk. |

| Grids | Nimrod/G | Commodity market | Nimrod/G enables users to define the types of resources needed and negotiate with the system for the use of a particular set of resources at a particular price [BA00]. This requires from the system to conduct resource discovery, which can become quite complex, as the numbers of re-sources can be large. Also, does the system need to support price negotiations, which may be complex as well. Both resource discovery and negotiation can become very cumbersome, if the users demand bundles instead of single re-sources. |
|-------|----------|------------------|---------------------------------------------------------------|

| Grids | OCEAN | Bargaining/ Contract net | OCEAN (Open Computation Exchange and Arbitration Network) is a market-based infrastructure for high-performance computation, such as Cluster and Grid computing environments [ACD+01] [PHP+03]. The major components of the OCEAN's market infrastructure are user components, computational resources, and the underlying market mechanism (e.g. the OCEAN Auction Component). In the OCEAN framework, each user (i.e. resource provider or consumer) is represented by a local OCEAN node. The OCEAN node implements the core components of the system, for instance a Trader Component, an Auction Component, or a Security Component. The implemented OCEAN auctions occur in a distributed Peer-to-Peer manner. The auction mechanism implemented in the OCEAN framework can be interpreted as a distributed sealed-bid continuous double-auction [ACD+01]. A trade is proposed to the highest bidder and the lowest seller. Afterwards, the trading partner can renegotiate their service level agreements. The renegotiation possibility one the one hand allows to cope with multiple attributes and with the assignment of resources to time slots. Nonetheless, the negotiation makes the results of the transparent auction obsolete. Neither the auction can enfold its full potential, nor can the negotiation guarantee to achieve an efficient allocation, as competition is trimmed. |
|-------|-------|-------|-------|

| Grids | Tycoon | Proportional Share | P2P clusters like the Grid and PlanetLab enable in principle the same statistical multiplexing efficiency gains for computing as the Internet provides for networking. Tycoon is a market based distributed resource allocation system based on an Auction Share scheduling algorithm [LHF04]. Tycoon distinguishes itself from other systems in that it separates the allocation mechanism (which provides incentives) from the agent strategy (which interprets preferences). This simplifies the system and allows specialization of agent strategies for different applications while providing incentives for applications to use re-sources efficiently and resource providers to provide valuable resources. Tycoon's distributed markets allow the system to be fault tolerant and to allocate resources with low latency. Auction Share is the local scheduling component of Tycoon. |
|---|---|---|---|

Table 1.1: Overview of market-based resource management systems (adapted from [Neu05, YB04])

In contracts to the presented market-based resource management system, the trading od the CATNETS project is divided into two layers, the application/service layer and the resource layer. On both layers, the participants have varying objectives which change dynamically and unpredictably over time. The following remaining sections of this deliverable show in detail what are the objectives of the project and in particular WP 1, and how this market is designed and realized.

## 1.4 Objective of WP1

The objective of working package 1 (WP1: Theoretical and Computational Basis) in this project is to engineer a market place for the CATNETS scenario of two connected markets. This can be achieved by designing (interdependent) market mechanisms for allocating the services among the agents and for allocating and scheduling the resources among the services in two different markets: the service market and the resource market.

The allocation can principally be conducted in a centralized, using economic resource brokers, or in a decentralized fashion, using self-organizing market mechanisms. The difficulty in designing such markets is that the underlying mechanisms through which the

participants act can have a profound impact on the results of that interaction [Jac02]. For instance, in a sealed bid auction the bidders simultaneously submit bids to the auctioneer without knowledge of the amount bid by other participants. In contrast, all bids under an open cry auction are available for everyone to see. Thus, in a sealed bid auction mechanism the participants do not learn as much about the valuations of the other participants as in an open cry auction. The higher information feedback may affect the bidding behavior of the market participants and could therefore lead to different outcomes. Furthermore, the market infrastructure (e.g. ways of communication) as well as the business structure (e.g. trading fees) can also influence the behavior of the participants and therefore the outcome of the mechanism [WHN03, Neu04a].

The Market Engineering approach chosen in this WP manages these influences by means of a structured, systematic, and theoretically profound procedure of analyzing, designing, evaluating, and introducing electronic market platforms [WHN03].

Appropriating the methodology of Market Engineering – which will be introduced in Section 2 – the following objectives constitute the outline of this report: In Chapter 3, the service market will be analyzed and the requirements a mechanism for this scenario has to fulfill will be defined. Furthermore, a centralized and decentralized mechanism will be designed which fit the defined requirements. In Chapter 4, the resource market will be analyzed and characterized, as well as the requirements for this market will be elicited. A central and decentral mechanism will be presented for the resource market scenario. Finally, Chapter 6 summarizes the contribution and gives an overview on future work.

# Chapter 2

# Market Engineering

In the general context of engineering, a design method refers to a way, procedure, or technique for solving an individual design problem[1]. These design methods can be either intuitive or discursive:

- *Intuitive approaches* involve creativity in the form of complex associations of ideas and aim at increasing the flow of ideas (e.g. using the Delphi method [PB84]). However, the results of intuitive approaches strongly depend on the designer's expertise, skills, and experiences. As such, intuitive approaches may fail to achieve suitable solutions for complex problems.

- *Discursive approaches* are strategies which decompose a complex design problem into several smaller, less complex problems. The design strategy intends to describe a step-by-step procedure to aid the designer in the matching of the unique problem situation along the overall design process with the available design methods.

The design of a market mechanism is a complex and interdependent task. Therefore, the approach of Market Engineering aims at the discursive, goal-oriented development of market institutions.

The Market Engineering Process Model is built upon the engineering design model proposed by [PB84]. In essence, the choice of the engineering design approach allows the explicit anchoring of two fundamental desiderata into the Market Engineering process model:

1. Utilization of economic models in the design process and

2. The use of behavioural and cognitive models to determine the needs and requirements of the potential customers and stakeholders.

---

[1]Parts of this text are taken from [Neu04a]

The first desideratum is very powerful, as it allows the integration of economic modeling into the Market Engineering process. Market Engineering hence is not neglecting prior work on the field of economic design, but can incorporate it into the process. Economic design is essentially concerned with social effects in markets derived from the analysis of abstract resource allocation mechanisms. Analogous to engineering design that deduces solution principles on the basis of physical effects, Market Engineering can make use of the social effects[2] to derive solution principles. The second desideratum is also of great importance in the design of market mechanism, as it primarily addresses social issues with many different customers involved. As such, the engineering design template is additionally enriched by elements of the service development process developed in marketing [SJ89].

## Process      Methods

| | |
|---|---|
| Operating Electronic Market | |
| Introduction | |
| Tested Electronic Market | ServQual |
| Testing | Labor-Experiments<br>Computational Game<br>Problem Solver |
| Preliminary Electronic Market | Blueprinting |
| Design and Implementation | Parametric Design<br>Mechanism Design<br>Primal Dual Programming<br>Morphological Design |
| Specification of the Requirements | |
| Environmental Analysis | Surveys<br>Interviews<br>SWOT Analysis |
| Formalizing of the new market service objectives and strategy | Joint Application Development<br>Quality Function Deployment |

*Upgrade and improve*

Figure 2.1: Stages of the Market Engineering Process

The corresponding Market Engineering process model and a selection of associated methods are outlined in figure 2.1 [WHN03] and briefly described in the following. The objectives and the strategy that governs the Market Engineering approach stand at the outset of the Market Engineering process. In the first stage – the environmental analysis – the requirements of the new market mechanism are deduced. In the second stage, the new market mechanism is designed and implemented. Having implemented the ap-

---

[2]Social effects are here defined very broadly as all regularities that depend on specific interpersonal interactions.

propriate market mechanism, it is tested upon its economic properties and its operational functionality in the third stage. Finally, the market platform is introduced.

At any stage of the Market Engineering process, there is a decision to be made whether to proceed with the next step or to repeat the prior one. The use of prototypes is again possible at any stage of the process, so that they are left out in the figure. The Market Engineering process not only structures the design process but also provides the designer with a whole array of methods that may support the individual sub-tasks. In the following each stage of the process is described particularly.

A detailed description of the single stages and the associated methods can be found in [Neu04a] and [Hol04].

## 2.1  Phase 1: Environmental Analysis

The environmental analysis stands at the outset of the market-engineering process. Basically, the term environment is used to describe the set of all individual circumstances in a market that are outside the control of the mechanism designer [Hur73]. Among others, these circumstances can be the number of potential participants, the characteristics of the participants (e.g., preferences, risk aversion), their social relations among each other, the characteristics of the resources, or the individual resource endowments [Smi89]. Since the performance of market mechanisms is inherently dependent on the underlying environment, the design of a market mechanism requires profound knowledge about the environment.

The environmental analysis is triggered by the specification of the objectives and the strategy of the designer. Objective of the environmental analysis stage is twofold: Firstly, the identification of a promising market segment for which a market mechanism is considered. Secondly, the analysis of the requirements potential adopters may have regarding the market mechanism. Corresponding with the engineering design process, this stage comprises two different phases: the environment definition and the requirement analysis.

### 2.1.1  Environment Definition

The central intuition of the environment definition phase is to pinpoint the environment for which subsequently the market mechanism is offered. The environment definition is clearly a marketing task. To systematize the phase more thoroughly, the environment definition is divided into three subsequent activities:

1. Market Definition,

2. Market Segmentation, and

3. Market Targeting.

Firstly, the relevant market is defined. In other words, the market designer is "carving out the arena in which it is going to compete for business" [FA03]. The market definition thus comprises information about the potential market participants, their geographical positions, their specific needs, and so forth. In short, the market definition[3] characterizes the demand-side for the market mechanism that is eventually determined by the definition of the trading object.

Secondly – having defined the relevant market – the designer divides the defined market into several segments. The division into segments is intended to disaggregate total demand into smaller pieces of demand. The smaller pieces of demand are ideally of homogenous nature such that it is easier to fine-tune the market mechanism to the needs of this market segment [FA03, Smi56].

Thirdly, the market segments are evaluated against each other. As a result, the target market consisting of one or more market segments is selected [HR86].

### 2.1.2   Requirement Analysis

Basically, the target market segment reveals the environment for which the market mechanism is intended. In order to gain potential agents as customers, the market mechanism must match with the particular needs of the agents. The requirement analysis phase consists of a thorough extraction of the potential customers' needs concerning the resource allocation problem and the environmental side-constraints [BCZ92, SSS00]. In other words, the requirement analysis seeks to describe the socio-economic environment consisting of the (potential) number of agents, their preference structure and risk attitude, the number of resources to be offered, their characteristics, and the agents' endowment. [Cra03] summarizes this as follows: "Good market design begins with a thorough understanding of the market participants, their incentives, and the economic problem that the market is trying to solve".

In summary, the result of the requirement analysis ideally comprises the following aspects: a description of the socio-economic environment (agents, resources, and preferences), the legal framework, and a requirement list what objectives the market mechanism must attain, and lastly what properties it must have.

## 2.2   Phase 2: Design and Implementation

The second stage – headlined as design and implementation – comprises the actual design process. It commences when the design problem has been sufficiently specified. Obvi-

---

[3]The terms market definition, segmentation, and targeting have been established in marketing for a long time. As such, these terms are also used in this report, although the definition of the term "market" is not exactly corresponding with the institutional definition given here. What marketing literature addresses with the term "market" is the environment, whereas the market mechanisms are left out.

ously, the aim of this phase is twofold. Firstly, the conceptual design of an appropriate market mechanism on the blackboard, and, secondly, its transformation into a running software system. Following the engineering design process, the design and implementation stage is decomposed into four major phases being the conceptual design, embodiment design, detail design and implementation

## 2.2.1  Conceptual Design

The second stage – headlined as design and implementation – comprises the actual design process. It commences when the design problem has been sufficiently specified.

The conceptual design step hallmarks the peculiarity that distinguishes the Market Engineering from the service development process. Essentially the design problem is abstracted in a way that the design object is abstracted to its functions. As the design object is the market mechanism, it has the function to allocate resources, provide the customers with information, enforce the allocation, and sue infringements and so on. The recourse to abstraction simply means "ignoring what is particular or incidental and emphasizing what is general and essential" [PB84].

Those functions are further divided into sub-functions reducing the overall complexity of the design problem. Then, the sub-functions are distinguished into important (i.e. main) or less important (i.e. auxiliary) functions. Important functions are tackled within the conceptual design phase, whereas the design of auxiliary functions is postponed to the embodiment phase.

Transferred to Market Engineering, the functions are also solved by means of social effects [Ore01]. Different than in engineering design, the entities that cause these effects are not form attributes of material, but a set of rules. For example, the designer may want to satisfy the function of an efficient resource allocation. In this case, the designer searches for market mechanisms that achieve this function.

The resulting abstract solution descriptions to all functions are aggregated into concepts. The concepts are furthermore supplemented by a calculation of profitability predicting the chances of the envisioned market mechanism in the competition. Finally, it is decided upon which concept – including abstract descriptions of the service enriched by profitability estimates – is further adopted.

## 2.2.2  Embodiment Design

While the conceptual design phase is concerned with the formulation of the problem and the search for abstract solutions, the embodiment design refines the abstract concepts to blueprints. A blueprint denotes a model that is more concrete than the concepts, but still independent of implementation details. In other words, the concept comprises at most a verbal description of the market mechanism. As such, many different blueprints

can be found that realize the same concept. During the embodiment design this verbal descriptions are transformed into a model with sufficiently low level of abstraction that traditional design techniques may be applied in order to implement it.

The main difficulty in embodiment design is that commonly "developers translate the subjective description of a need into an operational concept that may bear only the remote resemblance of the original idea" [Sho84]. As blueprints are, "[...] more precise than verbal descriptions of the service processes and therefore reduce ambiguity and the likelihood of misunderstandings that may originate from them" [HRRM00]. Furthermore, the technique of blueprinting almost prevents the designer from conceptual errors, because the blueprint allows " [...] the creation, study, and testing of services conceptually on paper before costly implementation" [HRRM00]. There are many methods that satisfy these four objectives of blueprints. The most common models that are presented in the context of blueprinting are flow diagrams or PERT (i.e. program evaluation and review technique) charts[4] [HRRM00, Sho82, Sho84]. The methodology Blueprinting tailored to embodiment design converting concepts into protocols is described in [Neu04b].

### 2.2.3 Detail Design and Implementation

The detail design phase starts out with the layout, which describes the central aspects of the system, but is still at a level that is not implementable. Detail design further refines the layout into a fully-fledged system model that is subsequently implemented. Apparently, this phase accounts for the software engineering effort in Market Engineering.

From the software development point of view, the precedent design phases of the Market Engineering process can be subsumed under the term requirement analysis. Different than the traditional sequence of interviews, or the use of Joint Application Development (JAD) meetings conducted by professional modellers, the Market Engineering process provides the designers with a systematic approach to collect design information from the experts. In other words, the precedent design phases of the market design process converts the activities of gathering, figuring out, and communicating what to build [HB95] into a closed discursive approach[5].

By doing so, the Market Engineering process supports the arguably most important step in the requirement analysis [DHJM99].

Once the designer has a clear idea how the market mechanism will look like (by means of the layout), an ordinary software development process can be started. State-

---

[4]Broadly speaking, PERT charts visualize tasks, durations, and dependencies among task. Each chart starts with an initial node from which the first task(s) originates. The tasks are represented by arrows, which indicate the identifiers of the tasks, the durations, the number of people assigned to them, and sometimes even the names of the employees involved. The arrow points at another node, which identifies the start of another task, or the beginning of any slack time. Related techniques are CPM or GANTT charts.

[5]The Market Engineering process thereby follows the four phases of the requirement analysis, conceptual design, logical design, validation and formal specification proposed by [BCZ92, Zmu83].

of-the art approaches like the V-model [Som01, TH93] supplemented by methods such as the FUSION [CAB$^+$94] or Coad/Yourdon [CY91] and tools such as UML [OF99, RJB99] are available such that the software engineering process will not further be elaborated.

Detail design is, however, more than software engineering – detail design phase is also concerned with the concretization of the business rules. Up to this point, only the key data concerning the business rules such as target costs and price ranges for the market mechanism exist. Once the properties of the market are clarified, reliable pricing schemes can be developed.

The end of the detail design and implementation is reached, when the market mechanism is fully implemented.

## 2.3   Phase 3: Testing

Having implemented the market mechanism, it is tested. Stage 3 denotes, however, not testing in general but the final acceptance test before the market mechanism is rolled out. The inclusion of a separated testing stage may also account for the case that the designer has sourced the implementation task out. The designer will then only accept the software system if it passes their acceptance testing. Apparently, the inclusion of a testing stage does certainly not exclude testing along the entire design process; it rather provides the designer with decision support whether or not the system can be launched in the field.

Nevertheless, the term acceptance testing is used here in a broad sense meaning all activities "used in quality control operations to decide between acceptance and rejection of production lots based upon an inspection of selected items" [MCM75]. What is tested is the software quality, and the quality of the service. While the former testing checks the functionality of the service system, the latter refers to the outcome of the market mechanism in economic terms (such as efficiency). Thus, the testing stage comprises two different testing phases before release: functionality and concerning economic performance testing.

When the market mechanism passed through the functionality (acceptance) test, it is tested concerning its economic performance. Now the question arises, how economic performance can be evaluated? This question is extremely difficult, as the outcome is determined by the behaviour of the agents and not by the mechanism. Milgrom summarizes the problems with behaviour as follows: "Behaviour is neither perfectly stable over time, nor the same across individuals, nor completely predictable for any single individual. Useful analyses must be cognizant of these realities" [Mil04]. In the discipline of economic engineering there are two approaches to address the evaluation problem of institutions:

- Axiomatic Approach

- Experimental Approach

The axiomatic approach imposes a couple assumptions upon human behaviour and calculates equilibrium strategies. With those equilibrium strategies, the performance of the institution can be calculated.

Alternatively, the experimental approach exposes humans to the institution, who will autonomously form their strategy. To make the laboratory experiment comparable, the demand and supply situation is induced to the participating human by means of a monetary incentive scheme. Hence, the performance of an institution depends on the real social interplay among the agents.

## 2.4   Phase 4: Introduction

After the pilot, the electronic market service can be introduced on a full-scale. Market engineering understood as the design of the institutional rules ends with the post-launch review, which measures the customer acceptance right after rolling out the service. Holistic market engineering continues, however. With the introduction of the electronic market service the operation cycle will be initiated.

# Chapter 3

# Specifications of the Service Market

## 3.1 Environmental Analysis

When designing a market mechanism, one has to be aware of the underlying environment for which the market mechanism is to be designed. Thereby, the environment description embraces information about the potential market participants, their needs, the characteristics of the traded resources, and their endowments. Once the environment description is available, the designer can elicit the requirements for the market mechanism from the potential participants of that particular environment [Neu04a].

Corresponding to the engineering design process, the design stage comprises two different phases: the environment definition and the requirement analysis.

### 3.1.1 Environment Definition

**Participants**

The market for trading services is spanned around basic services as sellers[1] and complex services as buyers:

**Complex Service**  A complex service is a modular software application which needs a set of basic service capabilities for fulfilling its goals. An internal logic translates the requirements of a complex service to a set or sequence of modular basic services.

**Basic Service**  A basic service is a module includable in a complex service.

For example, a complex service is part of a business service network providing PDF creations. This service needs a sequence of two basic services, a PDF creation service

---

[1]From a technical point of view, a service can be interpreted as an agent trading different services during a specific time span.

and a payment service. He combines them to a value-added business service.

The services in the form of sellers sell a set of basic services and are responsible for providing the traded basic services to the buyers and for procuring the required resources for the services on the resource market (cf. chapter 4).

Continuing our example, the PDF basic service (seller) is responsible for providing the PDF creation (in form of the PDF creation service) as well as the required resources (e.g. computation and storage service for the conversion).

The number of participants as well as the number of tradable services in the service market is difficult to determine. In practice, there exists no comparable market at the moment. However, related distributed information systems (e.g. Seti@Home, Gnutella) as well as large scaled markets (e.g. the XETRA market mechanisms of Deutsche Börse[2]) may be a clue for determining the number of participants and the number of different services:

**Seti@Home** Seti@Home was founded at the UC Berkeley and enables users to participate in the search for extra-terrestrial intelligence by analyzing data from the Serendip radio telescope in Puerto Rico[3]. Seti@Home is a distributed system with a central coordinator. Seti@Home has more than 5.000.000 people participating in the network and more than 10.000 active users per day [Set05].

**Gnutella** Gnutella is a Peer-to-Peer file sharing system which allows users the worldwide access and provision of information (e.g. music files). [AH00] analyze the network traffic and observe in a 24-hour period more than 35.000 connected peers.

**XETRA** XETRA (Exchange Electronic Trading) is a worldwide electronic securities trading system operated by the Deutsche Boerse Group in Germany. XETRA implements a centralized auctioneer and, in peak times, handles at the German Stock Exchange more than 500.000 orders a day [AG05].

Analyzing related systems and mechanisms gives a first impression of the number of participants and the number of tradable services. However, a detailed analysis has to be made in conjunction with the technical analysts within the simulation WP (WP2: Simulation) in the CATNETS project, as well as also complexity constraints of the simulation environment have to be considered.

Regarding the work done in WP2 (simulation), the simulator that will be selected for CATNETS must allow the simulation of Grids having up to 70 sites. This can be enlarged by an increasing number of agents per site (as also described in [Zin05] for the OptorSim simulator). Thus, the scalability requirements mainly depend on the simulator's capabilities. However, the objective is to scale the number of participants as high as possible.

---

[2]http://www.deutsche-boerse.de, 30/07/2005
[3]See `http://setiathome.ssl.berkeley.edu/` for details (accessed: 04.02.2005).

**Transaction Objects**

The products traded on the service market are completely standardized. There are no quality or capability differences between service instances of a specific service type. Quality of service (QoS) levels are modeled defining new service types. As such, an instance of a "gold"-QoS-level instance of a PDF creator traded once does not differ from a "gold"-QoS-level PDF creator instance traded at a later time. But, a "silver"-QoS-level PDF creator instance differs from a "gold"-QoS-level PDF creator instance.

## 3.1.2 Requirement Analysis

Having analyzed the participants and the traded products of the service market, the requirements which an adequate market mechanism for this environment has to fulfill can be elicited.

The theoretical basis for designing auctions has emerged from a part of game theory called mechanism design [Jac02]. A mechanism $\Psi$ specifies the available messages and the rules how to resolve it via clearing and price rules. Formally, a mechanism $\Psi$ is a pair $(\Phi, y_\Phi)$ where $\Phi$ is the language and $y_\Phi$ represents the resulting allocation $h$ and prices $p$. For any message profile $\phi \in \Phi$, the mechanism $\Psi$ computes the resulting allocation and prices as an equilibrium solution. Within the scope of practical mechanism design, it is the primary goal to investigate a mechanism that is applicable in certain situations and which attains an allocation that has desirable properties. As such, for tailoring an adequate mechanism for trading services in the Service Market, it is necessary that the mechanism accounts for the well-accepted properties an requirements from economic theory and mechanism design [Jac02]. Furthermore, the requirements stemming from the environment of the service market have to be considered.

**Requirements stemming from the theory of Mechanism Design**

The theory of mechanism design takes a systematic look at market mechanisms and their outcomes based on game theory [Jac02]. Within the scope of mechanism design it is the goal to investigate a mechanism that is applicable in certain situations. Achieving this objective, the designed mechanism has to fulfill the following general requirements [Jac02, Par01, Neu04a]:

**Allocative efficiency:** The mechanism should determine an efficient allocation. Assuming transferable utility among all participants, this is achieved if the total value over all participants is maximized. Allocative efficiency can be defined in an ex-post and ex-ante sense. Ex-ante efficiency takes preferences over expected allocations in consideration, whilst ex-post analyses preferences over realized allocations. A mechanism can only attain allocative efficiency if the market participants report their valuation truthfully. This requires incentive compatibility in equilibrium

**Incentive compatibility:** A mechanism is required to be incentive compatible. This is the case if all participants report their preferences truthfully. Participants may not have an incentive to untruthfully report their preferences in order to increase their individual utility. A mechanism is strategy-proofed if truthful revelation of the preferences is a dominant-strategy equilibrium.

**Budget balance:** The property of budget balance is concerned with whether the mechanism requires payments from outside the system or not. A mechanism is budget balanced if all payments made to the mechanism are redistributed among the participants. Neither funds from the system are removed nor is the system subsidized from outside. A weaker property is the concept of weak budged balance, i.e. net payments are made from the participants to the mechanism, but no net payment from the mechanism to the participants.

**Individual rational:** The constraint of individual rationality requires that the utility after participating in the mechanism must be greater or equal than before. Otherwise the participants would decide not to take part in the mechanism.

**Computational tractability:** Computational tractability considers the complexity of computing the outcome of a mechanism based on the agent's strategies [KP03]. With an increasing size of the message space, the allocation problem can become very demanding. Computational constraints may delimit the design of choice and transfer rules.

**Communication complexity:** Communication complexity considers the minimization of the amount of communication that is required to converge at a desirable global outcome of the mechanism [San99b]. With an increasing number of participants, it is required that the communication effort is minimized.

**Requirements stemming from the service market environment**

The requirements elicited from the mechanism design theory do, however, not fulfill all requirements for the service market in the CATNETS scenario. The underlying environment and its properties stem for more detailed and domain specific requirements as elicited in the following:

**Simultaneous trading:** The mechanism for the service market requires that multiple sellers (basic services) and multiple buyers (complex services) can trade simultaneously. This installs competition on both sides of the market.

**Immediate service allocation:** It is required that suitable buyer orders are matched immediately (or within a short time range) with suitable seller orders.

**No partial execution:** The mechanism is required to avoid partial executions of services. This is caused by the fact, that a partial execution of a service is useless for a buyer.

## 3.2 Meeting the Requirements – Related Work

The analysis of the service market results in a negotiation of a single attribute, the price for requesting one service instance, which fulfils additionally the three requirements from above. Using market-based control as a paradigm for controlling the service market leads to simple interaction of trading, i.e. buying and selling a service, among individual agents. Markets in general do not guarantee an optimal solution but they often achieve satisfactory results. First, related work of centralized mechanisms for resource allocation on the service market and second related work of decentralized economic resource allocation concepts are presented in the following sections.

### 3.2.1 Centralized Mechanisms

As shown in the environmental analysis (cf. Section 3.1), services can be characterized as standardized resources. In this context, the use of auctions is an efficient way to allocate these standardized resource, as well as to determine its prices. An auction is characterized as a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from the market participants [MM87].

At the moment, there are many auction schemas successfully applied in practice: for instance, Ebay[4] or Amazon[5] apply variations of English auctions for their trading platforms. Several double auction mechanisms were successfully applied in the financial domain (e.g. for trading stocks), the electricity domain, or the general B2B area. Thus, the application of auctions for trading services is deemed promising.

Treating auctions for standardized resources, single sided and double sided auction formats are usually analyzed and will be introduced in the following. Figure 3.1 represents a classification schema for a set of relevant auctions formats based on [WWW01]. For a detailed description of the capabilities and properties of these auction mechanisms, the reader is refereed to [WWW01] and [Neu04a].

**Single Sided Auctions**

Single sided auctions are mechanisms, where only buyers or sellers can submit bids or asks ($1 : n$ or $m : 1$ relations). The most prominent single sided auctions are the Vickrey Auction, the Dutch Auction, and the English Auction.

Single sided auctions are – from an economic point-of-view – well understood and applied successfully in different domains. However, single-sided auctions do not enable the simultaneous trading of multiple buyers and multiple sellers. Reviewing the defined

---

[4]http://www.ebay.com/
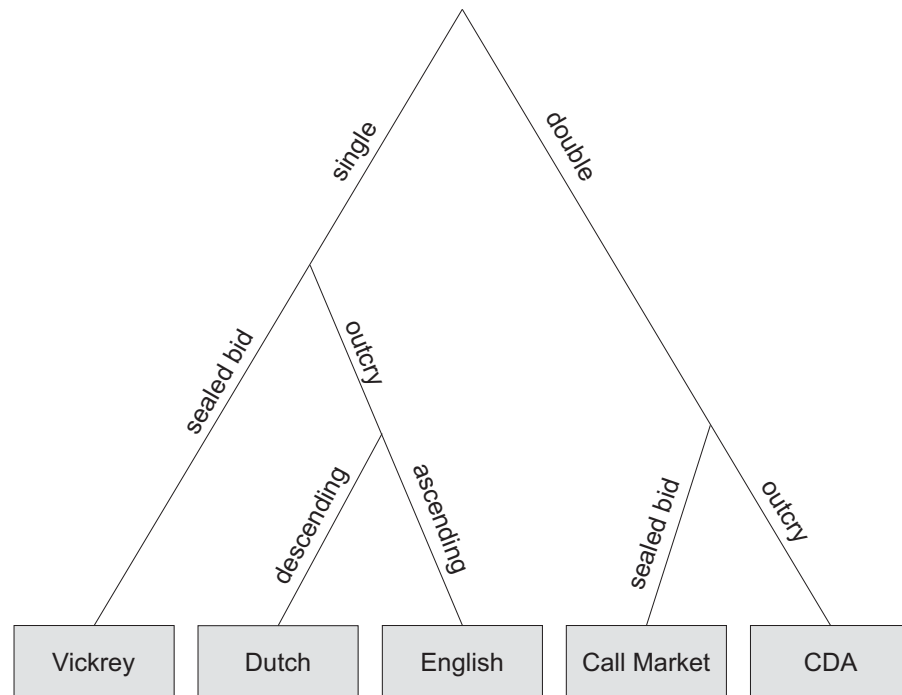[5]http://www.amazon.com/

Figure 3.1: Auction classification schema according to [WWW01]

requirements in section 3.1.2, these auction types cannot be applied for the service market in CATNETS.

**Double Sided Auctions**

Double-sided auctions or shortly double auctions are those auctions where competitive bidding takes places on both sides ($m : n$ relation). In comparison to traditional single-sided auctions, double auctions have received much less attention by modern economic theory.

For double auctions, where many buyers and many sellers compete against each other, it is difficult to game-theoretically model the strategic behavior of buyers and sellers [MM87]. Thus, theoretical models typically focus on the very simple two sided institutions, where agents engage directly in bargaining over the terms of exchange.

The review of the requirements elicited in section 3.1.2 implies, that the implementation of a double auction fits the defined requirements for the CATNETS' service market. Single sided auctions have the drawback to be either provider- or consumer-oriented and thus asymmetric. Double auctions enable providers and consumers to offer bids simultaneously and thus, the application of a double auction for the service market is deemed promising. Furthermore, the requirements stemmed from the theory of mechanism design (cf. Section 3.1.2) can be (approximate) fulfilled by a double auction:

It is possible to design an (approximate) efficient[6], budget balanced, individual rational, and (approximate) incentive compatible double auction mechanism. Furthermore, the computational and communication complexities of a double auction are manageable in large-scaled markets. For instance, the XETRA system at the German Stock Exchange also implements a (modified version) of a double auction and handles more than 500.000 orders a day [AG05].

Furthermore, the requirements coming from the environment of the service market (i.e. simultaneous trading, immediate execution, no partial executions) are also fulfilled by a double auction.

## 3.2.2 Decentralized Mechanisms

The centralized market uses a double auction on the service market, which is widely accepted in the auction theory and practice. Most of the current research is done in the centralized and hierarchical approaches, because the decentralized approach comes along with a loss of system control and thus has problems with widely acceptance. Therefore, the following description of related work describes fractions, which have to be brought together to get the economic behavior of von Hayek's vision.

The predictability issue is the main problem with a central control. No omniscient controller exists in the real world. Contrast to the centralized auctioneer, there is no need of any agents in the system to know all the parameters of the system in order for the system to function smoothly. A market-based system necessarily is easy to expand and maintain since there is no need for a central coordinator.

All complex systems share common dynamical behaviors. These possible behaviors are: stability, transient instability, persistent oscillation, deterministic chaos, and random motion, which in turn are characterized by parameters of the system [Cle96]. Depending on the values of the parameters the system shows one or several of these behaviors. The parameters characterize the communication paths between the entities in the system, the efficiency of the communication, the time delays for communication, the accuracy of response and the overall response of the entities, and the expectation of the agents.

In the decentralized mechanism the market-based control emerges from the interaction of individual goals of the agents rather than having a central goal imposed from above. Friedrich August von Hayek [HBKC89] and other Neo-Austrian economists understood markets as decentralized coordination mechanisms, as opposed to a centralized command economy control. In addition to macroeconomic thoughts, Hayek's work also provides concrete insight into the working mechanisms of economic coordination. However, a formal description of this self-organizing market mechanism does not so far exist. The Catallaxy concept is based on the explicit assumption of self-interested actions of the par-

---

[6][MS83] show that there cannot be any double auction (exchange) which is efficient, budget balanced, and individual rational at the same time. However, [PKE01] show that the efficiency of an exchange can be approximated while cleaving on the budget balance and individual rational properties.

ticipants, who try to maximize their own utility and choose their actions under incomplete information and bounded rationality [Sim57]. The term Catallaxy comes from the Greek word katallatein, which means to barter and at the same time to join a community. The goal of Catallaxy is to arrive at a state of coordinated actions, through the bartering and communicating of members, to achieve a community goal that no single user has planned for. The main characteristics of the Catallaxy [Hop99] are enumerated below. Each property imposes several requirements upon the design of an information system embodying a Catallactic approach.

1. Participants work for their own interest to gain income. Every system element is a utility maximizing entity, supports means to measure and compare income and utility, and to express a desire to reach a defined goal.

2. Participants can only estimate the effect of action alternatives on an income or utility maximization goal, as nobody has total knowledge and foresight of the environment. Instead, "constitutional ignorance" of the rationally bounded participants makes it inevitably impossible to know the exact environment state. For large and very dynamic information systems, this observation leads to a design shift. Instead of trying to overcome this limitation by central means, e.g. through synchronization of the system by introducing round-based brokerage, the focus shifts to improving the computational intelligence of the actions to decide under uncertainty, and to adapt to constantly changing signals from the outside.

3. Participants communicate using commonly accessible markets, where they barter about access to resources held by other participants. The development of prices for a specific good, relative to alternatives, and whether they are increasing or decreasing, leads buyers to look for alternative sources of procurement and thus enhances the dynamics of the market. In that view, a market is simply a communication bus; not a central optimization component, or a mechanism or a protocol.

Decentralized mechanisms, like in most file sharing networks, e.g. Gnutella [AH00] or Kazaa, have no central point to collect supply and demand before matching. Each client decides for himself which service provider to match to based on technical parameters like estimated download time. APPLES [CO00], WINNER [AF99] or MARS [GAR96] are other examples for localized control. The problem of localized control is the missing assurance on allocation stability [WJB03].

All these complex system perform their resource allocation using technical metrics without market-based control.

In the decentralized architecture an iterative bilateral negotiation protocol, similar to a contract-net, is used as no complete information is available. Both agents approximate to the trade-off point in iterative steps exchanging offers and counter-offers. This process is described as monotonic concession protocol [RZ94].
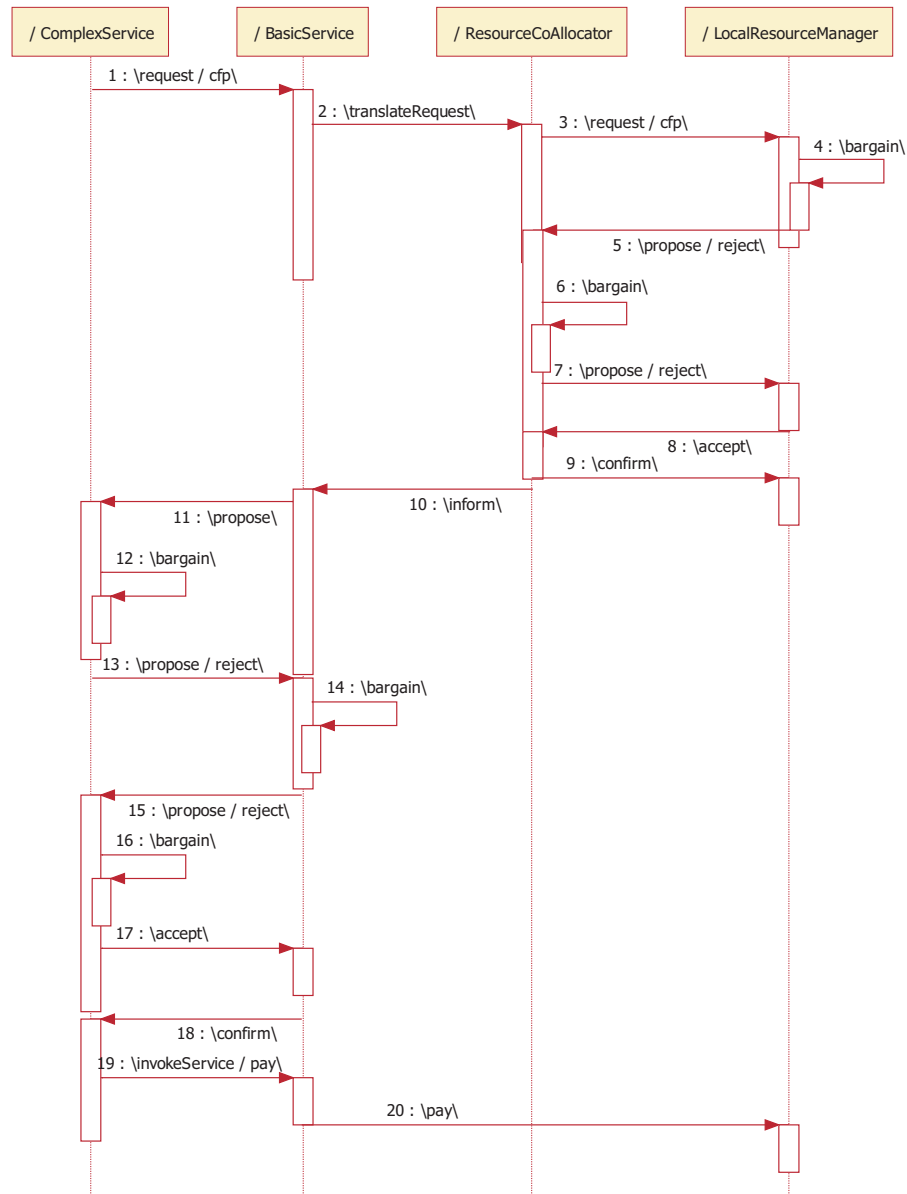
Figure 3.2: UML sequence diagram of the negotiation between the market participants.

A preliminary model of our protocol is shown in figure 3.2. The participants on the service market are the complex service and the basic service, and on the resource market the resource co-allocator and the local resource manager. For clarity, the basic service is split in its buyer and seller side: basic service (seller side) and resource co-allocator (buyer side).

The complex service requests a basic service, according to his process demand. The basic service translates this request to resource layer and the resource co-allocator and starts the negotiation with several local resource managers. The local resource manager analyzes the request and creates an offer and this process iterates until an agreement (accept) or reject is reached. If an accept is reached, the resource allocator confirms and informs the basic service about the contracted resources. The basic service continues the negotiation with the complex service, using the information from contracting the resources. The negotiation on resource layer is processed only once. It is impossible to renegotiate a resource contract. The resuming negotiation with the complex service uses the same negotiation protocol and after an accept the payment process is initiated which pays the basic service and the resource. A reject on the service market will lead to a reject ton the resource market.

However, the communication sequence shows only one possible scenario for the description of the negotiation protocol: mostly, several prospects are imaginable when receiving a proposal by an opponent: The agent could reject the proposal, accept it or send a counter-offer. This decision should include past and forecast experiences. The number of alternatives describes the complexity of the decision process; the more alternatives exist, the bigger the search space, which increases the quest for a suitable solution. Though, this may not exceed the time frame that is accepted by the opponent for the reply message/counter proposal. According to [Pre98, LW00], four mechanisms/strategies for negotiation can be differentiated:

1. Rule-based mechanisms are subject to the premise, that all possible states are completely known and the environment remains static during processing. This might be suitable for a market with fixed catalogues, as re-consulting a catalogue with static prices and articles will not change the decision rules. Double auctions, however, cannot be matched, as implementing all decision alternatives when changing prices or articles' attributes is impossible. The conceived market model of CATNETS will inherently comprise dynamic and complex states of the environment; therefore rule-based mechanisms are not appropriate.

2. Argumentative mechanisms include not only the proposal for prices but also purchase-supporting arguments which extend the negotiation dimensions. The arguments highly depend on the application, thus no standards can be set and this implicitly leads to a higher complexity in decision and modeling. Furthermore, it exhibits a deviation of the single dimension negotiations.

3. Game-theoretic approaches assume the market situation to be a multi-stage game

between buyer and seller. The strategy results from the analysis of the negotiation problem. For the formulation of a result, the availability of an offer of the counterpart is not necessary; the analysis can rely on prospects. The internal model of the agent comprises a calculus which explicitly includes all probable behaviors of the opponents. Multi Agent Systems can be implemented using Game-theoretic approaches, but due to the high computation prerequisites, these systems may be limited to a number of 20 agents [MTE03]. Thus, Game-theoretic approaches are not suitable for the concerned market model, due to the scalability concerns.

4. Heuristic-adaptive approaches assume incomplete knowledge a priori, and therefore they also expect defective decisions. Agents adapt their strategy by relating the behavior of the market and their own activities [SF89, SG89, CB98, BKS00]. The counteroffer of the opponent is contemplated as feedback of the former own proposal. As the whole spectrum of opponents' proposals cannot be anticipated completely beforehand, the strategy uses forms of "trial and error" to formulate offers. An easy example of use is shown in [Pre98] where heuristic rules are combined with easy learning rules. Each agent is willing to negotiate a price that is below (buyer) respectively above (seller) its own price limit. His autonomous decision is to determine a price for selling/buying. The market mechanism is a round-based continuous double auction, and all agents use the same implementation. The implemented heuristics determine the target price of the agents in their negotiation steps. A number of numerical or boolean parameters determine the strategy; these variables will be adapted during the lifetime of an agent. In MAS learning methods like neural networks [Fau94], Q-Learning [SC95], classifier systems [Hol92], Bayesian networks [Nea96] and evolutionary algorithms [Gol93] can be identified. This implementation is typical for realization of heuristic-adaptive strategies.

From the mentioned mechanisms, heuristic adaptive strategies show the most scalable behavior [Eym03] and are favored for adoption in large MAS, that are addressed in CATNETS. To enable an adaptive behavior of the participant a learning mechanism is necessarily required. The following section introduces the depicted learning algorithms and evaluates them for adoption in CATNETS.

**Adaption of strategy**

The combination of artificial intelligence and machine learning has become increasingly complex in the last years. Brenner [Bre02] classifies learning methods in non-conscious learning, routine-based learning and belief learning. However, non-conscious learning processes are discussed not to appear in experiments. They are used in well-known situations which stands in contrast to the CATNETS approach. Belief learning focuses on the individual and not on the whole population. For global optimization, belief learning is not to be suitable. For bilateral negotiation processes, optimizing global behavior of a population, routine based learning strategies are most applicable. Routine based learning

strategies describe the learning process on population level. It is acceptable, to model the learning process not granularly on the individual level, as the emerging behavior of the complete system is in focus of CATNETS. There are mainly two possible ways how to implement a reinforcement learning strategy: the Roth-Erev model and evolutionary algorithms. The Roth-Erev [ER98] model is a quite simple model, which is adequate for a small, fixed set of actions and strategies. Because of being limited to a small, pre-defined set of actions, the Roth-Erev model is not applicable of CATNETS.

Evolutionary algorithms are able to deal with a very large set of actions and strategies and allow the sets of strategies increase endogenously. Therefore this type of algorithms is more applicable to CATNETS. Nevertheless, both models should not be considered to be more than quite crude approximations on a population level of real conscious learning processes.

In economic simulations lots of research efforts on evolutionary algorithms can be found. We selected the STDEA (Smith Taylor Decentralized Evolutionary Algorithm) for CATNETS [SNT98]. This algorithm showed good results in the predecessor project Cat-Net. Other possible strategies are numerical optimization procedures. Eymann and Mller [MTE03] could show, that results in similar learning mechanisms showed very similar results. The STDEA is a decentralized evolutionary algorithm, which means that it has no global evaluation metric (fitness value), which is used in Genetic Algorithms [Gol93] to separate the under performing participants. A fundamental quality of the mechanism is the decentralized communication and fitness evaluation, using locally available data. Every agent sends a plumage object after a successful transaction, advertising its average income (fitness) and its genes (genotype) to all agents of the population after an evaluation phase, i.e. after it has carried out a certain number of negotiations with this genotype. If an agent receives a plumage object from other agents, it decides using a blindness probability, whether the plum-age object is evaluated, avoiding premature unification of the genotype. Sender and recipient remain anonymous. If a certain maturity threshold of received plumage is exceeded, the agent replaces his old genotype with the evolved version after the completion of evaluation, selection, recombination and mutation phases as in normal genetic algorithms. The mutation rate is also influencing the algorithm, which determines the frequency and the extent of explorative behavior of the population. A detailed explanation of the learning algorithm is presented in section 3.3.3.

When comparing with other, numerical optimization strategies [PT02] and decentralized learning, it must be admitted, that e.g. Powell's algorithm as well as the simplex method provide better results than the STDEA, because evolutionary algorithms per-form a routine based learning, which constitutes a slower learning process than observed in reality. However, the substantial advantage of optimization strategies to decentralized learning mechanisms becomes obvious if the size of the population is varied: In the case of one single agent, the numeric algorithms take advantage of their directed search, in contrast to the random exploration of the decentralized learning mechanisms. Increasing the size of the population the optimizer will reach its performance limits, whereas the learning mechanisms do not lack scalability and even perform better with an increasing

number of agents.

A mixed model is OVID (Optimized Variation-Imitation-Decision) [Bre96] [MTE03]. The OVID model presents an option of combining the advantages of the genetic algorithm STDEA, the numeric optimization procedure of the simplex method and the imitating and directed exploring behavior of human cognitive processes. This algorithm does not depend on a constant information flow between the agents, but can meaningfully optimize its own behavior at any time. It is thus more robust than STDEA and yields better results than pure numerical optimization approaches. The OVID model is currently only evaluated in a test bed. There are no results using this algorithm in real application. Therefore, it is considered to be a risk for the project using this model for learning. In CATNETS the use of the STDEA algorithm is recommended, which has proved to be able to handle an elevated number of agents in simulation and prototype.

### 3.2.3 Summary of Related Work

Summarizing the related work results in two concrete recommendations for the centralized and decentralized implementation of the Service Market:

The central mechanism for the Service Market will be a (continuous) double auction as it fits the requirements defined in section 3.1.2. The decentralized mechanism will comprise a bargaining concept on base of heuristic adaptive strategies and the STDEA algorithm.

## 3.3 Design of the Service Market

Application Layer Networks (ALN) encompass heterogeneous resources by a high number of geographically distributed devices and administrative domains, which are logically coupled together for providing processes on application level. This comprises computational services, data services and mixtures of these services. We expect ALNs to be shaped by lots of basic services that can be dynamically combined to value-added complex services (like in Service-oriented architectures [SH05]. The orchestration and customization of these basic services can be understood as an inherent service, that must be accomplished by the network as well, due to the complexity and the expertise requirements which must be hidden from the application. Basic services thus offer the interface accessing computational resources for complex services, while hiding the orchestration and implementation details.

In Service-oriented computing, users define complex business job requirements, which are then broken up into collections or sequences of basic services that together provide the desired functionality. The market participants, complex service as a buyer and basic service as a seller, have various objectives, tasks, strategies and demand patterns, which might change dynamically and unpredicted during life time.

Traditional approaches, using centralized policies require complete state information which is not available in dynamic and complex networks [KBM02]. As an acceptable system-wide performance matrix is impossible to define, we use an economics-based paradigm derived from human economies for the management or resource allocation and orchestration.

These models are based on exchanging and acting on price signals. The participants work for their own utility; thus, they evaluate the signals received and act according to some utility function, which predicts an utility increase out of the effect of that action. On a system-wide scale, a bird's eye view on such market dynamics shows continuous matching of demand and supply, a case of emergent coordination.

This section will provide a detailed presentation of the market models on the service market. Basic design issues like bidding language and message specification, which are the same on both mechanisms, are defined in subsection 3.3.1. The centralized market mechanisms are specified in subsection 3.3.2 and section 3.3.3 presents the decentralized market.

### 3.3.1 Bidding Language and Message Specification

A bidding language specifies, how agents have to submit their bids and, thus, which messages the agents exchange. For the service market, the bidding language can be formalized as follows:

Let $\mathcal{N}$ be a set of $N$ buyers (complex services) and $\mathcal{M}$ be a set of $M$ sellers (basic services), where $n \in \mathcal{N}$ defines an arbitrary buyer and $m \in \mathcal{M}$ an arbitrary seller. Furthermore, there is a set of $G$ discrete services $\mathcal{G} = \{g_1, \ldots, g_G\}$ with $g_k \in \mathcal{G}$.

A buyer $n$ can express the valuation for a service $g_k$ by $v_n(g_k) \geq 0$, i.e. the maximum price for which the buyer $n$ is willing to trade. The reservation price of a seller for allocating a service $g_k$ is denoted by $r_m(g_k) \geq 0$[7], which represents the minimum price for which the seller $m$ is willing to trade.

Subsequently, the order $B_n(g_k)$ of a buyer $n$ for a service $g_k$ can be represented as:

$$B_n(g_k) = v_n(g_k) \tag{3.1}$$

and the order $B_m(g_k)$ of a seller $m$ is formalized as

$$B_m(g_k) = r_n(g_k). \tag{3.2}$$

For instance, a complex service $n$ wants to buy a PDF service $g_k = \{PDF\}$ and values this service with $v_n(PDF) = 4$. The formalized order of the buyer $n$ is represented as

$$B_n(PDF) = \{4\}.$$

---

[7]It is also possible to mark seller orders by a negative sign.

A counterpart for the buyer $n$ could be a basic service $m$ offering a PDF service $g_k = \{PDF\}$ with a reservation price of $r_m(PDF) = 3$. This is formalized as

$$B_m(PDF) = \{3\}.$$

For the implementation of the service market and the resource market, the service level agreement language WS-Agreement will be used. WS-Agreement is a specification defined within the Global Grid Forum (GGF). It defines a language and a protocol for advertising the capabilities of service providers (in our case of basic services) and creating agreements based on creational offers, and for monitoring agreement compliance at runtime [ACD$^+$05]. The use of WS-Agreement is in line with the developments and specifications done in WP 3 (Proof-of-Concept).

The presented bidding language for the service market is a small subset of WS-Agreement and can be mapped to it. The mapping from an existing WS-Agreement to the bidding language is, however, not possible as some parts of the WS-Agreement specifications are not used in the service market.

The following XML encoded document depicts a WS-Agreement encoded offer for the above described offer for a PDF service:

```
<wsag:AgreementOffer>
    <wsag:Name>
        Offer
    </wsag:Name>
    <wsag:AgreementContext>
        CATNETS
    </wsag:AgreementContext>
    <wsag:Terms>
        <wsag:All>
            <wsag:ServiceDescriptionTerm wsag:Name="exceutable"
                    wsag:ServiceName="PDFService">
                <job:arguments>/some/file/to/convert</job:arguments>
            </wsag:ServiceDescriptionTerm>
            <wsag:GuaranteeTerm wsag:Name="ReservationPrice">
                <wsag:ServiceScope>
                    <wsag:ServiceName>PDFService</wsag:ServiceName>
                </wsag:ServiceScope>
                <wsag:BusinessValueList>
                    <wsag:CustomBusinessValue>
                        <catnets:reservation>3</catnets:reservation>
                    </wsag:CustomBusinessValue>
                </wsag:BusinessValueList>
            </wsag:GuaranteeTerm>
        </wsag:All>
    </wsag:Terms>
</wsag:AgreementOffer>
```

The valuation and reservation prices are not part of the standard WS-Agreement specification. WS-Agreement permits, however, the extension of the specification via domain

specific schemas. Thus, the prices are included as as domain specific schemas as a subset of the `CustomBusinessValue` tag using a separate XML-schema.

### 3.3.2 Centralized Market

As discussed in section 3.2.1, a double auction institution fits the specified requirements. In the following, the basic design principles of a double auction institution for the CAT-NETS service market will be sketched.

**Double Auction Market**

In a double auction market, 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 homogenous goods. An order book is responsible for storing non executed orders of the agents. For instance, in the CATNETS scenario there will be $n$ different order books, each for one of the $n$ different services.

Figure 3.3 depicts the high level architecture of the service market for CATNETS. Complex services can submit buy orders to the order books; basic services can submit sell orders. Each set of homogeneous services (e.g. PDF creator services) is traded in a single order book.
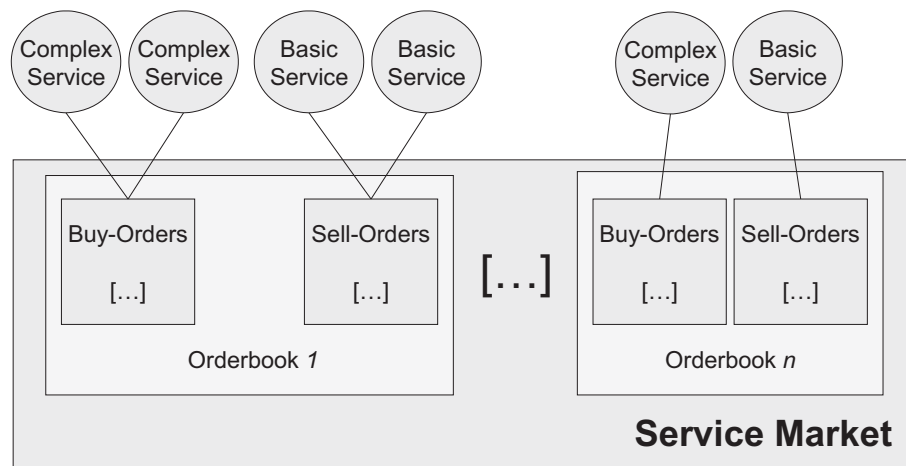


Figure 3.3: The service market including several double auction order books.

**Winner Determination and Pricing**

Buyers and sellers in the double auction submit their bids in a sealed envelope to the auctioneer. The auctioneer aggregates the bids to form supply and demand curves. This is done by calculating the number of participants who want to buy or sell at a specific price.

Suppose there are $4$ buyers with bids for $p(\cdot) = a$ and $2$ buyers with bids for $p(\cdot) = a + 1$. Aggregating these demands results into a curve expressing that there are $6$ buyers who want to buy for $p(\cdot) = a$ and $2$ buyers who want to buy for $p(\cdot) = a + 1$.
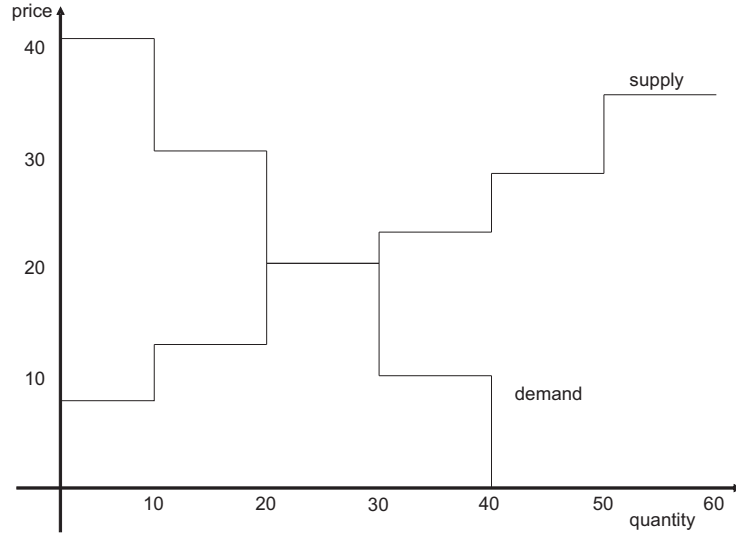


Figure 3.4: Demand and supply curves in a double auction.

Once these curves are aggregated, they are used to set a specific price for trading – the price at which supply equals demand in this case. Figure 3.4 shows exemplarily aggregated supply and demand curves for a specific service $S_i$. For instance, 10 of the buyer orders want to buy the service $S_i$ for a price which is less or equal than $p(S_i) \leq 30$.

Now suppose the demand and supply curves given in figure 3.5. In this case, there is a price tunnel between 18 and 20. Any price within this range will be acceptable because the supply and demand curves overlap.

This overlap is resolved using the $k$-pricing schema. The parameter $k$ sets the fraction of the profit that goes to the buyers and the sellers [PMN05]. Having a parameter $k \in [0, 1]$ given, a market clearing price $p = (1-k)a + kb$ is chosen from the interval $[a, b]$ confining the range of all possible market clearing prices. The interval denotes the price tunnel. In this case, the interval is given with $a = 18$ and $b = 20$.

Usually, a balanced order book is assumed, i.e. the number of buyers equals the number of sellers. Therefore, in literature $k$ is set to $k = 0.5$ as it privileges neither the buyers' side nor the sellers' side [Fri91]. Thus, for the CATNETS application, $k$ is also initially set to $k = 0.5$[8].

---

[8]It may be adapted during the simulation of the market by a configuration file.
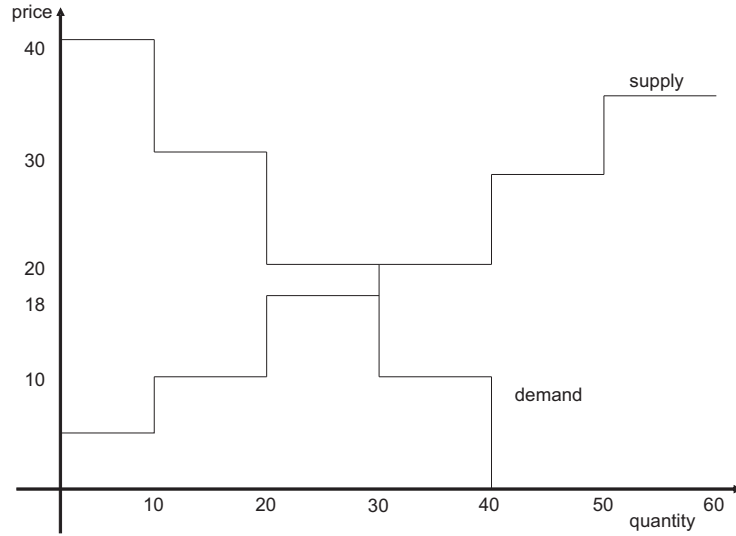
Figure 3.5: Demand and supply curves in a double auction with a price tunnel.

**Order Types**

In literature, a variety of order types for double auctions can be found [FSSW02]. For instance, order types implementing basic logical constraints to allow traders to control the price or the timing of their bids. Moreover, there exist special order types including more intelligent logical constraints to adjust price dynamics in electronic markets [KM05].

The two most common order types are the limit order and the market order [Nab96]. The limit order with a price $p(\cdot) = x$ indicates, that a participant wants to buy for at most $p(\cdot) = x$ or that a seller wants to sell for at least the price $p(\cdot) = x$. Furthermore, if a trader submits an order for a service at the prevailing price, the order is called market order (best execution).

In order to comply with the agent strategies of the decentralized market, the service market will only consider limit orders, i.e. a direct bid of the valuation of an agent.

**Continuous and Periodic Trading**

A key consideration in double auctions is the timing of the clearing process, i.e. the timing of determining the auction winners and thereby the allocation of the resources. Double auctions can be either cleared continuously (Continuous Double Auction) or periodically (Periodic Double Auction, Call Market):

A Continuous Double Auction (CDA) is a double auction where buyers and sellers simultaneously and asynchronously announce bids and offers. Whenever a new order enters the market, the auctioneer tries to clear the market immediately. Thus, the CDA is

advantageous especially in terms of immediacy.

A Call Market is a double auction with periodic uniform clearing, e.g. the auctioneer clears the market every fives minutes. All orders in a period are collected in an order book and will be cleared periodically. Assuming none time-critical resources, the call market is advantageous in terms of enhancing the overall welfare in a market.
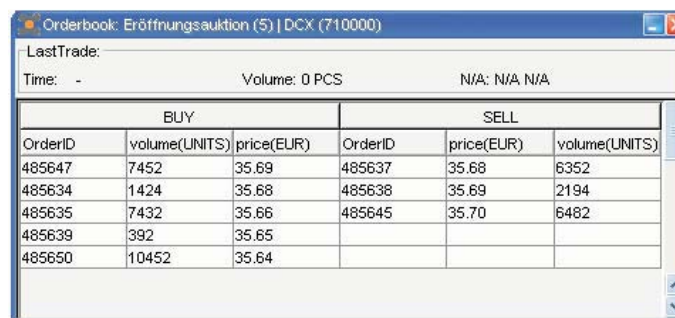
To meet the requirements specified in section 3.1.2, both, the continuous double auction and the call market with a short time period fulfill the simultaneous trading of multiple participants, an immediate service allocation, as well as avoiding partial executions.

A short time period may increase the overall welfare of a market; considering the immediate service allocation, a continuous clearing would be superior. As both clearing concepts can be easily implemented into a software system, the final decision concerning the clearing of the market will be made during the simulation runs[9].

**Implementation**

There exist several open source implementations of the $k$-double auction implementing a CDA and a call market which can be used for the CATNETS project. For instance, JASA[10] (Java Auction Simulator API) implements various variants of the double-auction as well as components for experiments, agents and learning algorithms. JASA is designed to be light-weight and high-performance. The double auction implementation makes use of the scalable 4-heap algorithm presented in [WWW98].

Another interesting implementation of a double auction is done within the Meet2Trade platform[11]. Meet2Trade offers several components for a generic auction design and order book management. Figure 3.6 shows exemplarily the order book of a Meet2Trade double auction market.



Figure 3.6: Meet2Trade order book.

The market mechanism for the service market will implemented as a reusable compo-

---

[9]This can be simple achieved by a configuration setting.

[10]See http://www.csc.liv.ac.uk/~sphelps/jasa/ for details (accessed: 09.10.2005)

[11]See http://www.meet2trade.org/ for details (accessed: 09.10.2005).

nent for the simulation environment. As such, the decision whether to adapt JASA or the double auction implementation of Meet2Trade will be made during the integration of the auction into the simulator.

### 3.3.3 Decentralized Market

The market in the decentralized approach is mainly a communication bus where self-interested agents barter for standardized basic services (see section 3.2.2 for the economic background and related work). Compared to the centralized market approach using an auctioneer, we do not have the possibility to send the bids to a central instance. Therefore, we need to process a search at first, that sends the demand of a complex service to possible basic service entities. In a second step, the bargaining between one complex service (buyer) and basic service (seller) follows which determines the price for the good.

The requirements for the search at the service market are quite simple. The search mechanism has to search only for the specified service type. No additional constraints have to be taken into account. The selection of the search algorithm depends on the computational efficiency regarding the required network bandwidth and optional reorganization steps needed to adapt the changing environment. Therefore, we can use any existing, widely accepted search methods like flooding or DHTs.

**The CATNETS market**

After the search for services, the selection process of a basic service is performed. Compared to the centralized market, this selection process fundamentally differs. The global order books of the centralized matchmaker become local order books of each complex service which covers all offers of one basic service demand. If a complex service requests a sequence of basic services, a complex service has an order book for each basic service demand. Figure 3.7 shows the local order books of several complex services.

A local order book collects the bids of the basic services and puts them into an order, beginning with the cheapest price. As described in the centralized market, the standardized basic services don't change over time. Only the price differentiates the basic service of one specific type. This leads to a single attributive negotiation on the service market.

The change from a global order book (centralized approach) to local order books of every complex service (decentralized approach) changes also the supply-demand curves. In the decentralized market we only have a supply curve, because many suppliers (basic services) submit bids for one demand. A possible supply curve shows figure 3.8. The reservation price represents the upper price limit of a buyer.

At the end of the search step we have a list of possible basic service suppliers, which can be incomplete. We assume, that the supply is "good enough" for the following selection process. A complete list of all possible suppliers is hard to realize, because the
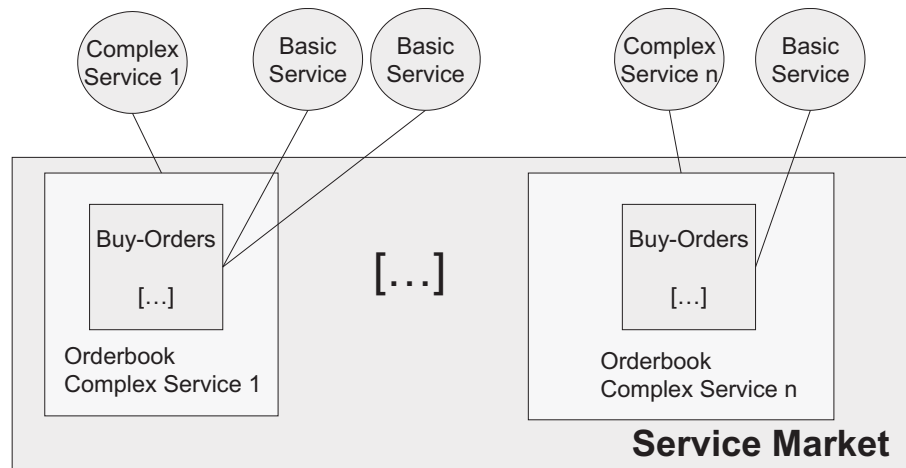
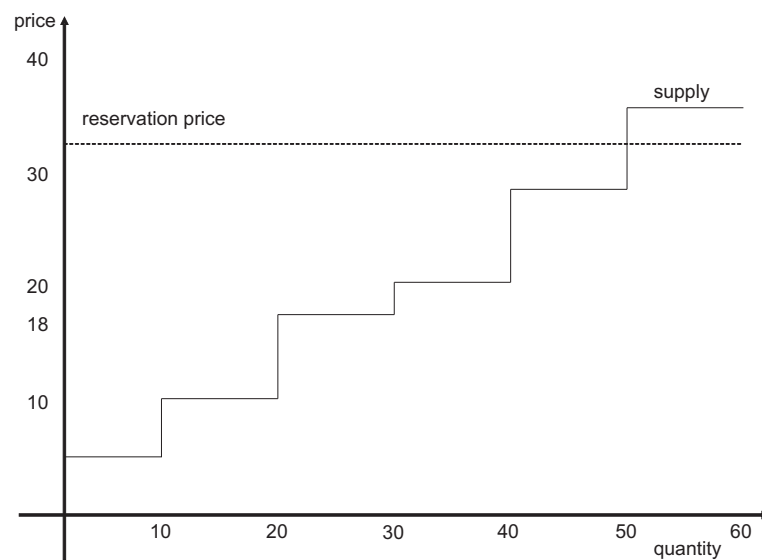Figure 3.7: The service market including several complex service order books.



Figure 3.8: Supply curve in a bilateral negotiation.

assumed dynamic behavior changes the state of the system often and the costs for a complete list become too high.

**Bargaining and Pricing**

Starting point for the winner determination through bartering is the cheapest basic service offer. The complex service selects the cheapest bid and starts the bargaining with the selected basic service provider. The definition of a strategy for winner determination through bartering is essential in the decentralized CATNETS market. The initial situation is described like depicted in figure 3.9. A (human) principal defines an indifference price that equals his estimation about the value of the good. For a buyer, this is a maximum price, for the seller a minimum price. So, the utility gain equals the amount between price of the purchase and the indifference price. The start price represents the price where the strategy begins to negotiate. By agreeing concessions, the opponents come closer to the middle and a possible contract. A transaction is unlikely, if the closure zone is empty, which might result when indifference prices do not build an overlapping zone.



Figure 3.9: Bilateral negotiation process.

The goal of the software agent strategies is maximization of their own utility. For a buyer, this means an maximization of the distance between the indifference and purchase price and for the seller the enlargement of the interval between the seller's purchase price and his indifference price. The proposed realization for the CATNETS market is the usage of a heuristic factor, that decides on the percentage change of the negotiation's starting price in the following, successful completed transactions. The higher this parameter is, the lager is the aspiration towards the enlargement of price distances (a detailed explanation

presents the next subsection). The agent is not aware of his reached level and there is no optimal goal. The attempt to maximize his own utility is not limited by any restriction regarding the absolute value nor time constraints and is a never ending process.

The continuous aspiration, to be better than other agents, is also influenced by the selected learning algorithm. The search for a good parameter configuration which assures good utility gain, leads to comparison with and re-combination of known configurations of other agents. This results in the decentralized, evolutionary learning algorithm STDEA [SNT98].

**Negotiation protocol**

The negotiation protocol for the CATNETS project defines the interaction schema between the negotiating parties. This is a sequential process of exchanging communicative acts. If there is complete market information like in the centralized market, the negotiation protocol is short and quite simple. In the decentralized market model, the point of complete information is never reached; both agents converge to a tradeoff point in an iterative way using the exchange of offers and counter-offers. Rosenschein and Zlotkin ([RZ94]) call this a monotonic concession protocol.

The negotiation protocol messages are sent between the agents in a bilateral way. Figure 3.10 shows the bargaining protocol on the service market. The negotiation process on the decentralized CATNETS market is divided into different phases. The first part of the negotiation is the request for a specific basic service. As we described above, the buyer (complex service) specifies his demand and the seller (basic service) creates a proposal for this demand. The second phase is the selection of one basic service with which the negotiation continues. Afterwards, in the third step, both bargain until they reach an agreement or possibly fail. In the final phase of the negotiation protocol, the agents confirm their negotiation outcome.
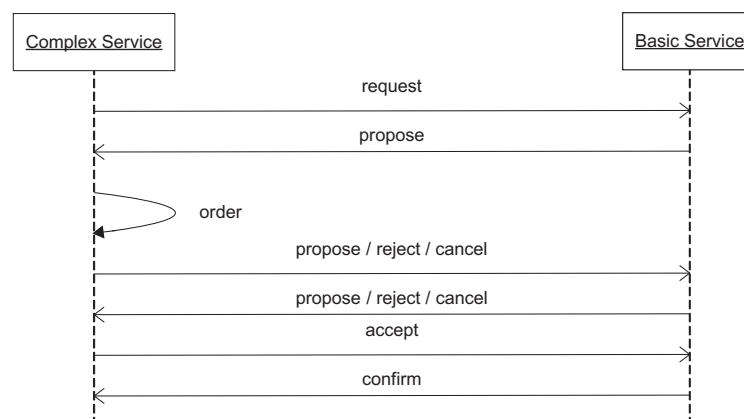


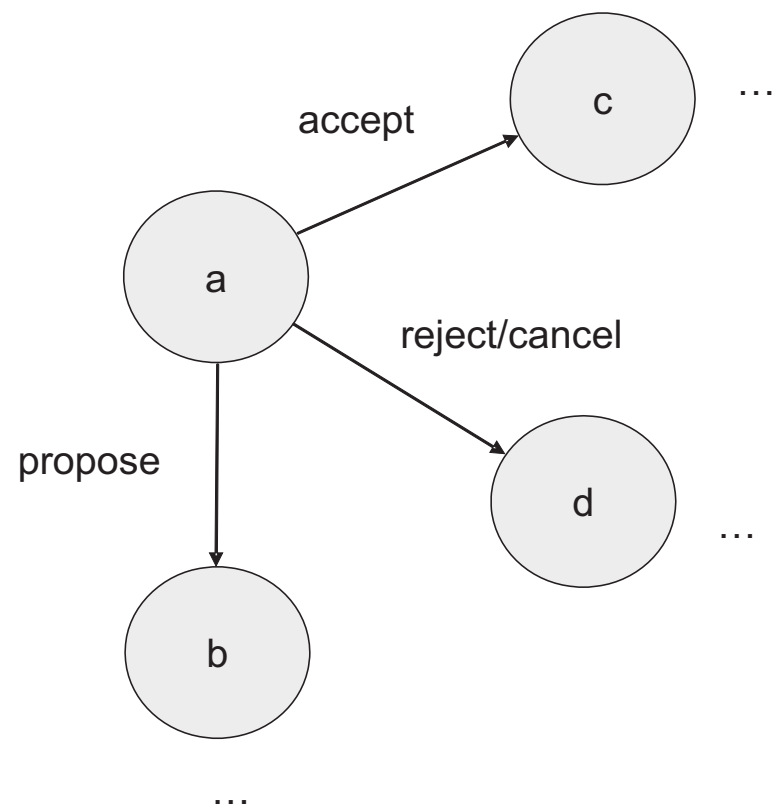Figure 3.10: The bargaining protocol on the service market.

Figure 3.11: The states of the negotiation protocol.

The negotiation protocol has different states, where decision about the next communicative act has to be made. If the negotiation protocol is in state a (see figure 3.11), the strategy has three possible decision options: accept the offer, propose a counter-offer or reject the offer. A detailed description of the decision-making process presents the next subsection.

Compared to the centralized communication protocol, where there are two types of message, bid and ask, the decentralized bargaining protocol needs five different message types: request, propose, accept, reject and confirm. The semantic meaning of the message payload is described using the proposed bidding language (see section 3.3.1).

The decentralized market model combines the protocol, the heuristic strategy and the the learning algorithm to specify a market-based control mechanism for resource allocation in complex and dynamic systems. The next section gives an overview, what now have the proposed concepts impact to the other work packages which use and evaluate these concepts.

**Negotiation strategy**

The negotiation strategy is used to reach the utility goal. The strategy specifies the actions, an agent uses for attainability of his goals. He chooses the best alternative action for his goal on the basis of information about the environment. The negotiation strategy, which is described here, is based on the AVALANCHE strategy [Eym00]. The strategy defines 5 basic parameters: *acquisitiveness*, *priceStep*, *priceNext*, *satisfaction*, *weightMemory*.

These are used in the bargaining strategy and are explained in the following section. Additionally the message format of the strategy component is defined with 5 parameters:

**Factory-ID** The Factory-ID specifies the negotiated good, which is on the CATNETS service market the standardized basic service.

**Proposal type to generate** The second parameter defines the type of proposal to generate. This can be either a bid or an ask. This distinction is important on the one hand to rate the level of the own bid in ratio to the market price and on the other hand to determine the direction in which a concession can be made.

**Opponent's current price, My old price** These two values define the price signalling, which have to be related to each other. The "Opponent's current price" is the signalled price of the negotiation partner, which represents his current offer. "My old price" is the last price sent to the negotiation partner, which is the base value for making concession.

**Conversation-ID** The parameter "Conversation-ID" is mainly a technical parameter which is used to identify the negotiation, when simultaneous negotiations are running. At the CATNETS service market, this parameter is not important, because for simplicity we have not implemented simultaneous negotiations.

**Current market price**  The current market price of the negotiated good is needed to evaluate the transferred prices of the negotiation partner. The price is signalled using a price distribution. If the price distribution is not defined yet, like at the beginning of the negotiation, a new price distribution is set. After the initialization of strategy three non-null values exist: the own price, the opponent's price and the market price. In the following we distinguish between the first-time generation of the offer price and the succeeding negotiation steps. The difference is, that there is no value of a former offer at the initial negotiation step.

**First-time generation of an offer**  If the negotiation just started, an agent can use only two of the the three possible clues: the bargaining price of the opponent which he receives with the demand and the subjective market price. He can use only these prices to generate the first counter-offer. A buyer-bid is generated like this:

```
if (proposalTypeToGenerate == Proposal.ASK) {
  currentPriceDistribution.setInitialPrice(
     currentPriceDistribution.
        getLastAgreementPrice() *
     (1 - myGenotype.getPriceNext()),
     proposalTypeToGenerate);
     currentPriceDistribution.setLimitPrice(
        currentPriceDistribution.
           getWeightedAverage(),
        proposalTypeToGenerate);
```

These code lines define the closure zone of the bargaining. The initial price – this is the starting price of the negotiation – is computed using the last price, a successful former negotiation, that has ended (lastAgreementPrice), less a winner price gain (myGenotype.priceNext).

**Adaptation of the initial and limit prices**  The parameter myGenotype.priceNext modifies the particular starting value of a new negotiation. The result of the last negotiation is saved and it is esteemed as quite sure, to achieve the almost the same value as in the last negotiation. An achievement of the last would not increase the utility of the agent. Therefore, he tries to buy cheaper the next time and decreases his initial price with a winner discount. The multiplication with myGenotype.priceNext computes the starting price for the next transaction.

It is assumed that an agent is never satisfied with an achieved goal and wants to achieve a higher outcome at the next negotiation. Either he decreases his concession rate, or he begins with a higher starting value (as a seller) so, that he reaches a higher outcome with the same concession rate. Like humans adapt their behavior to new situations, an agent is

able to change these parameters using an evolutionary learning algorithm . Calculational, this proceeding increases the starting price of the seller (computed as the sum of indifference price and the desired utility increase) and decreases the starting bid of the buyer (computed as the indifference price less desired utility increase) respectively.

**An example for priceNext=0.15:** A buyer agent bought a good for ■100 the last time. Using his last outcome and *priceNext* parameter, he will start the following transaction with ■85. Now, the question about how to choose the limit price arises. In the present case, the average of the known market price (currentPriceDistribution.weightedAverage) is used. Another option can be, that the first seller offer is reversed. Here, the initial price is the last closure price increased with a winner bonus. Using this price, the seller decreases his price in the negotiation step by step until he reaches his limit price:

```
} else {
    //proposalTypeToGenerate == Proposal.BID
  currentPriceDistribution.setInitialPrice(
  currentPriceDistribution.getLastAgreementPrice() *
    (1 + myGenotype.getPriceNext(),proposalTypeToGenerate);
  currentPriceDistribution.setLimitPrice(
  currentPriceDistribution.getWeightedAverage(),
                proposalTypeToGenerate);
}
```

After the agent has defined the closure zone, it must be checked, whether the exchanged bid of the opponent is in the closure zone already and a negotiation needn't be started. In this case, there is no negotiation object needed and the offer price can be accepted immediately.

```
if ((proposalTypeToGenerate == Proposal.ASK) &&
  (opponentsCurrentPrice <= myNewPrice)) {
    // I am buyer, so if initial offer
    // from opponent is already lower
    // than I am willing to buy at, accept
    recommendedAction = ACCEPT;
    this.executionState =
      "ask was underbid on first round";
    myNewPrice = opponentsCurrentPrice
} else if (
  (proposalTypeToGenerate == Proposal.BID) &&
  (opponentsCurrentPrice >= myNewPrice)) {
  // I am seller, opponent is buyer,
  // and he buys for more than I want to get
```

```
        recommendedAction = ACCEPT;
        this.executionState =
            "bid was overasked on first round";
        myNewPrice = opponentsCurrentPrice
    }
```

This was the last step of the first negotiation round. The next paragraph describes the handling of the negotiation history.

**Storage of the negotiation's executions state**   The negotiation history is stored in an object, to save the last communicative act. In the parameter history.priceStep the level of the possible concession for the following negotiation is computed and stored using the initial prices of the negotiation partners. A precentral computation of the concession level out of the current prices leads to a infinite convergence to a concession price without reaching this price exactly. We assume in the following, that a negotiation data-set exists and the own and the last opponent's bid is known.

**Adaptation of the market price using signalled price information**   The first part of the strategy is the adaption of the subjective market price for the negotiable good using signalled price information. How the price information are used to create a market price, every agent has to decide on himself, because there is no central information store like a database or an auctioneer. Every price information, which is signalled to an agent, is associated with exponential first order smoothing of the existing price information and stored in the current price distribution (market price) object. The weighted mean value of the method "getWeightedAverage()" states the current estimation of this market price.

The rating of the offer due to a subjective market price through an agent is absolutely needed for the operativeness of the strategy. If such a rating is not done, the agent starts at every bid a negotiation and tries to finish this negotiation. Therefore, this opens the floodgates to speculative bids. An example:

Anna can ask ∎1000 for a good, Benno's price of the good is ∎10. They negotiate until they close an agreement at about ∎505. If the market price of the good is ∎900, Benno bought the good very cheap. If the real market price is at ∎20, Anna's bid was profiteering. This cannot be taken as granted; it is very likely, that one of the agents has signalled knowingly wrong information. Only the knowledge of the real market price can prevent, that one of the two agents is discriminated.

This consideration is based on the presence of common value goods: The market price measures the price of the good, because it can be sold infinite times, in principle [San96]. The more information an agents gets about different prices, the better is his market picture and the better is he able to arrange out if it his own pricing (or the rating of the others pricing).

**Protection against usury bids**   The buyer checks the initial signalled price before he starts a negotiation, whether this price is in an interval which can lead to a successful negotiation result. He compares the signalled price with price bids from proceeding negotiations with other agents. This is independent from a successful result of these negotiations with other agents. In particular, he is able to reject usury bids at the beginning. In CATNETS, bids which are higher than double market price are rejected immediately.

**weightMemory**   The weighting ratio of current price information and historic price information influences the strategy of the agents largely. The higher the weighting factor (implemented as myGenotype.weightMemory) is, the faster the market price adapts to the current market situation. But, the agents can be influenced faster through short-time price fluctuation on the market. This will emphasize the characteristic behavior of the agent. A "correct" value of the weighting factor cannot be defined a priori. The value arises from the cooperation of the agents. The parameter weightMemory is predefined at the initialization of the agent. It influences the learning in the following way:

```
if (opponentsCurrentPrice > 0) {
   currentPriceDistribution.addValue(
      opponentsCurrentPrice,
      myGenotype.getWeightMemory());
   this.executionState = "updated streetPrice = " +
      currentPriceDistribution.getWeightedAverage();
   }
```

**Check about the obtainment of a tradeoff price**   The next step is the determination, whether the opponent signalled a acceptable bid. It is checked similar to the direct acceptance of a bid in the first round as described above. The current bid is compared about intersection with the own last price (myOldPrice).

```
if ((proposalTypeToGenerate == Proposal.ASK) &&
   (opponentsCurrentPrice <= myOldPrice)) {
   // I am buyer, opponent is seller,
   // and he sells for less than I would pay
   recommendedAction = ACCEPT;
   this.executionState = "ask was underbid";
   double myNewPrice = opponentsCurrentPrice
   currentNegotiationHistory.setMyLastPrice(
      myNewPrice);
   currentNegotiationHistory.setOpponentsLastPrice(
      opponentsCurrentPrice);
   currentPriceDistribution.setLastAgreementPrice(
      myNewPrice);
```

```
     historyHash.put(conversationID,
        currentNegotiationHistory);
     priceHash.put(factorID, currentPriceDistribution);
     return myNewPrice;
     }
```

If there is a intersection, the bid is accepted and the status is updated. The negotiation history is also updated (currentNegotiationHistory). The currentPriceDistribution-object, that contains the market price, stores information about future concession rates in the lastAgreementPrice parameter. The process for acceptance is laterally reversed and thus is not described here in detail.

**Abort of the negotiation due to dissatisfaction with the previous development**   If the last negotiation steps didn't reach a result, we are in the middle of a negotiation and know our own bid and the opponent's bid. The next step estimates, if it makes sense to continue the negotiation and if there is a chance to come to an agreement. First check is, whether the opponent signalled a usury bid which is higher than 100% over the current estimated market price. Continuing this negotiation would have less promise on success and the stored market price in the following negotiation rounds distort. Therefore, the bid is rejected.

```
double upperBound =
   currentPriceDistribution.getWeightedAverage() *
      ACCEPTABLE_SPREAD;
double lowerBound =
   currentPriceDistribution.getWeightedAverage() /
      ACCEPTABLE_SPREAD;
if ((opponentsCurrentPrice >= upperBound)
   || (opponentsCurrentPrice <= lowerBound)) {
   // unconditional constraint: usury;
   // opponent's price is 100% off the street price
      recommendedAction = REJECT;
      this.executionState = "rejected: usury";
      myNewPrice = opponentsCurrentPrice;
      currentNegotiationHistory.setMyLastPrice(
         myNewPrice);
      currentNegotiationHistory.
         setOpponentsLastPrice(
            opponentsCurrentPrice);
      historyHash.put(negotiationID,
         currentNegotiationHistory);
      priceHash.put(factorID,
```

```
        currentPriceDistribution);
    return myNewPrice;
    }
```

The next paragraphs show the strategy of evaluating offers using the parameters satisfaction and acquisitiveness. 3.12 gives a schematic overview of this process. First, it is checked whether the former offer was better or not. In a second step the parameter satisfaction and in a third step the parameter acquisitiveness is evaluated.
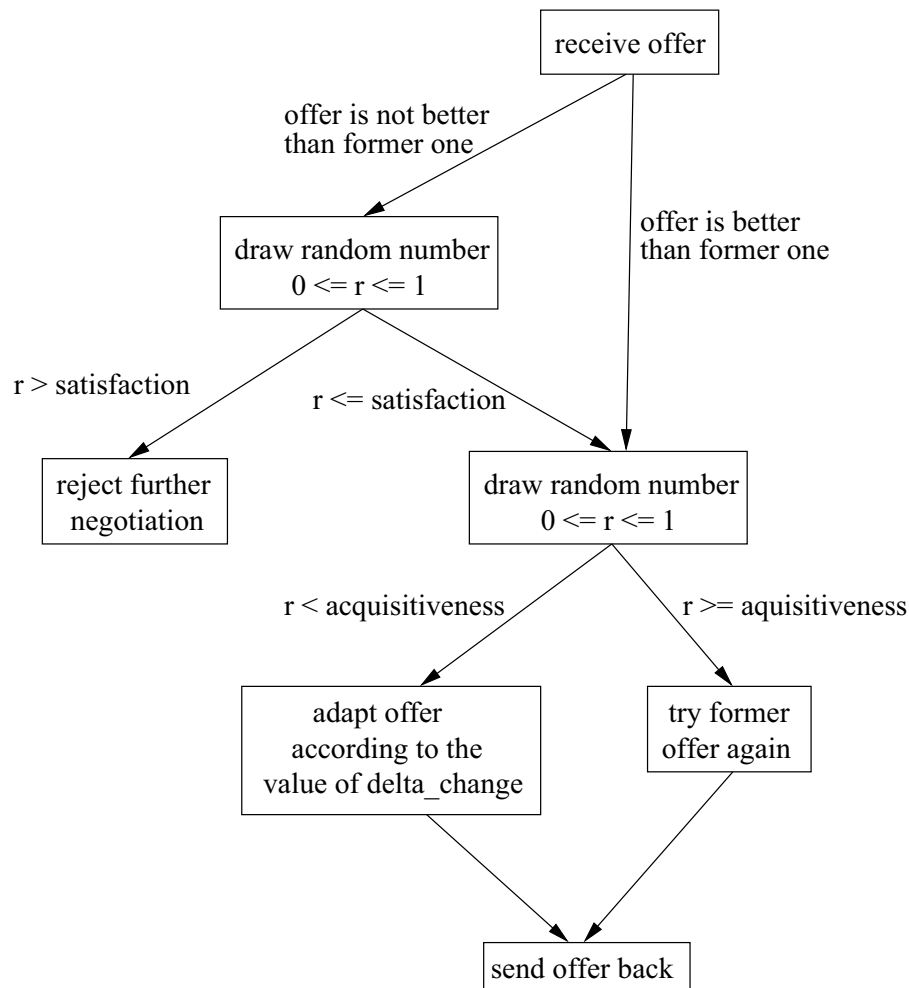


Figure 3.12: The evaluation process for received offers.

**Satisfaction** If the offer price is higher than the market price, but less than double market price, an agent has to decide whether to continue. It is possible that the market price of the negotiated good increased his price. To simulate a more complex deterministic "probe", the stochastic factor myGenotype.satisfaction is checked which has values between 0 and 1. A satisfaction value of 0.75 means, that the agent is satisfied to 75% with

the negotiation process and continues. An agent with satisfaction 0 will abort a negotiation at once and an agent with satisfaction 1 will never abort a negotiation.

**Commitment to concessions**   Third, the agent checks if the opponent has made an concession. If the negotiation rounds do not satisfy the agent (i.e. The negotiation partner didn't change the price compared to the last round and therefor shows a competitive behavior), a stochastic check against the satisfaction parameter about aborting the negotiation is done (i.e. a reject instead of a counter-offer is sent):

```
if (RND.simulate() >
    myGenotype.getSatisfaction()) {
        recommendedAction = REJECT;
        this.executionState =
            "rejected because of dissatisfaction";
        myNewPrice = opponentsCurrentPrice;
```

Both, the buyer and the seller agent, can abort a negotiation at every round – the shorter the negotiation, the higher is the chance for a successful transaction. A simple statistical computation shows, that the chance of a successful negotiation decreases with the negotiation time. Making concessions only makes sense in combination of the time pressure.

**Decision about making concessions – the parameter acquisitiveness**   The parameter Genotype.acquisitiveness defines the probability to make an uniliteral concession in the following negotiation. The real decision is determined with a stochastic "probe". The value of the parameter does not determine the action, but sets a specific probability. The value interval is between 0 and 1. A value of 0.7 means a probability of 70%, that an agent follows a competitive strategy not making concessions. A seller agent with a true boolean value will not adapt his price and will signal the same price to his opponent.

```
// third strategy component:
// will we make a concession
this.executionState =
    "positively calculating concession";
    double concessiveness = RND.simulate();
    if (concessiveness <
        myGenotype.getAcquisitiveness()) {
        //no concession, repeat our last proposal
        recommendedAction = PROPOSE;
        this.executionState =
            "no concession, repeat price";
```

If the boolean value is false, the agent computes an new price, adapting the price using his concession parameter. A buyer agent will rise his offer, a seller agent will lower his price. An agent with acquisitiveness value 1 will never change his price and an agent with acquisitiveness value 0 will make an unilateral concession, adapting his price towards the opponents price.

**Decision about the concession's level – the parameter priceStep**    The concession's level (priceStep) is defined at the beginning of the negotiation with the parameter Genotype.priceStep during the initialization of the negotiation history:

```
history.setPriceStep(
   Math.abs(opponentsCurrentPrice - myNewPrice) *
      myGenotype.getPriceStep());
```

The definition of an absolute concession level does not take into account the level of the demanded price which leads to an implementation with a percentage computation. The direct indication of the concession as a percentage of the bid discriminates the seller, because he assumes an absolute value during the negotiation. The following example points up this problem:

Anne would like to sell her good for ■80, Benno offers only ■60. If both, Anna and Benno, decide on a 20% concession, Anna submits a bid with ■16 less in the next round and Benno submits a bid with only ■12 higher. The average between the crossing bids of ■64 (Anna) and ■72 (Benno) is ■68. The difference from the starting utility is for Anna ■12 and for Benno only ■8. The utility gain is therefore not symmetric. Only Benno will benefit under the assumption that the same prices mean identical utility change.

With such a process, the market price decreases over time. That would lead to a wrong picture of the market-based coordination mechanism. Thus, a percentage of the difference between the initial starting prices of both parties is introduced in the strategy which does not change during the negotiation. A value of Genotype.priceStep = 0.25 means a computation of the concession level as 1/4 of the first stated difference. If both opponents negotiate in the same way and make concession to each other, they meet each other on the half way in the third negotiation round under the assumption of no negotiation abortion. An example for explanation:

Anna submits ■80 as her first offer, Benno offers ■60. The difference between the two offers is ■20. Both negotiation partner have priceStep = 0.25. The concession level is ■20 * 0.25 = ■4 for each negotiation partner. In the second negotiation round, both decide on concession; Anna offers now ■75 and Benno ■65. In the third round again, both decide on concession and they meet at ■70.

The computation of the concession's value for a buyer is presented below. He increases his demand price and makes a concession for a seller. The limit price is the buyer's upper bound:

```
recommendedAction = PROPOSE;
double delta\_price =
   currentNegotiationHistory.getPriceStep();
   if (proposalTypeToGenerate == Proposal.ASK) {
      double myNewPrice =
      Math.min(
         currentPriceDistribution.
            getUpperLimitPrice(),
         myOldPrice + delta\_price);
```

The seller follows the same process respectively. If no other condition is fulfilled, the last offer is submitted again:

```
this.executionState =
   "no condition applied, repeat last price";
   double myNewPrice =
      currentNegotiationHistory.getMyLastPrice();
```

This heuristic strategy is used to realize market-based control on the decentralized CATNETS market. The strategy is supported using the Smith Taylor Distributed Evolutionary learning Algorithm (STDEA) for faster adoption to the dynamic environment. In section 3.2.2 we gave an overview of possible learning algorithms. We chose the STDEA algorithm, because there were good results in the former CatNet project with this learning component. The following paragraph will give short introduction the basic setting of the STDEA.

**Learning using STDEA**   The learning algorithm in CATNETS is necessary for the adoption of the strategy to the changing environment. It is assumed, that the parameters which define the strategy can be stored in an object. The learning algorithms receives this object with the current parameters as input and returns the new parameters which can be used in the strategy. Important methods of the learning interface are:

- getRecommendedLearningAction(): This method gives feedback to the strategy about what to do with the learning information. Two options are possible in CAT-NETS: Either ignore the learning information or substitute the existing strategy values with the new ones.

- getGeneration(): getGeneration() provides a counter for a successful learning process. This counts the generations of the evolutionary algorithm.

- createFitnessInformation(): This creates the fitness information for learning actions.

The basic concepts of STDEA are already presented in the related work section. We will describe here the algorithm in detail. The STDEA is an evolutionary algorithm. There are three information entities in this algorithm which we separate in the following:

- Plumage: This entity is used for transportation of successful strategy parameter combinations (genotypes).

- Gene: A gene describes one parameter of the strategy. Two different gene types are possible: a boolean and a float gene.

- Genotype: A genotype subsumes the single genes to a strategy parameter set.

**Specification of a gene and a genotype**  A gene represent one characteristic attribute of an agent's strategy. The gene has to provide some basic functionality which are derived from genetic processes in biology:

- replicate: The replica method creates an identical copy of the original parameter value.

- mutate: The mutation-method alters the current value of the parameter a small step. If we have boolean gene, this process will negate the value. The float gene is always a parameter between 0.0 and 1.0. We will use a creep mutation of the float gene which means, that we adapt the parameter with small steps of maximal 0.01. This prevents jumps of the parameter's value.

The genotype is a set of genes. The genes of an agent represent in the CATNETS project the strategy parameters. Therefore the agent has a set of 5 genes described at the strategy section. The genes in CATNETS are: acquisitiveness, satisfaction, PriceStep, PriceNext and weightMemory. An agent cannot change these values himself. They are stored in a genotype-object which enables a change through mutation. The initialization of a genotype is processed like the following:

```
public class Genotype extends Vector
   implements IMutateable, IRandomizeable,
      Cloneable, Serializable {
   public Genotype(
      double p_acquisitiveness,
      double del_change,
      double del_jump,
      double p_satisfaction,
      double w_memory,
      double p_reputation) {
         this.add(new FloatGene(p\_acquisitiveness));
```

```
        this.add(new FloatGene(del\_change));
        this.add(new FloatGene(del\_jump));
        this.add(new FloatGene(p\_satisfaction));
        this.add(new FloatGene(w\_memory));
   }
```

The whole vector is able to perform a mutation and crossover operation with another genotype. A mutation changes every parameter with a probability of 5%; this changes the bit at the boolean gene and slowly modifies the float value at the float gene.

```
public final void mutate(Random rand) {
    for (Enumeration e = elements();
       e.hasMoreElements(); ) {
          Gene g = (Gene) e.nextElement();
          if (rand.nextFloat() < 0.05) {
             g.mutate(rand); }}}
```

**Initialization of the genotype using a configuration file**  The CATNETS agents are initialized with a random distribution of parameters which are stored in a configuration file. The random distribution has fixed values for the mean value and variance on agent population level. During the setup the agents are initialized using these values. This enables a heterogeneous distribution of bargaining strategies and an individual genotype of every agent.

```
<strategy>
   <acquisitiveness>
      <average>0.60</average>
      <variance>0.0</variance>
   </acquisitiveness>
   <price\_step>
      <average>0.35</average>
      <variance>0.0</variance>
   </price\_step>
   <price\_next>
      <average>0.20</average>
      <variance>0.0</variance>
   </price\_next>
   <satisfaction>
      <average>0.50</average>
      <variance>0.0</variance>
   </satisfaction>
   <weight_memory>
```

```
        <average>0.50</average>
        <variance>0.0</variance>
    </weight_memory>
</strategy>
```

The distribution function interprets a variance of 0 as taking the mean value every time. A variance of 0.2 with a mean value of 0.5 leads to an interval from 0.3 to 0.8 for the concrete value assuming a equal distribution.

**Measurement and propagation of fitness information**    Learning and adoption demand the declaration of a reference value almost at all times which should be achieved or exceeded. Information about the performance of other market participants is needed to learn a successful market strategy. The performance has to be measurable as well as comparable. In CATNETS we use the fitness information of an agent as a performance metric.

The fitness of every bargaining strategy results from the income level, that an agent obtains from actions leading to successful transactions. This income is added to the agent's wealth. We assume, that every agent starts with the same wealth. This makes the income increase of the agents comparable at discrete time levels. All buy and sell actions have influence on his actual balance. An agent has to pay for his acquired goods on the one hand and on the other hand he increases his balance with the sale of goods.

In CATNETS, the agents can use the market place for free. This means, the communication and computing costs do not influence the strategy. An ex-post analysis of these costs is processed in the evaluation work package (see the deliverable of WP 4 for further information). An agent whose capital is equal to zero, is removed from the market place. In simulation this means, that the agent will be deleted, in the prototype this agent is sent back to the owner's agent pool. His strategy is not available any more. This concept enables a consistent performance measurement in simulation and prototype.

Every agent creates after a successful transaction a plumage which contains his fitness and his genotype. The fitness of an agent is the difference between the market price and the subjective value of the traded good, i.e. the surplus achieved in the negotiation. The plumage is transferred to a randomly selected agents.

```
double profit =
  myCurrentProposal.getUnitPrice() -
     strategy.getAveragePrice(factorID);
  sendMessage(
     new GroupMessage(
        "inform", holdingID,
        (Plumage)
        strategy.getLearningModule().
           createFitnessInformation(profit)));
```

After an agent receives the inform message, he stores the plumage. The plumage number in the storage is counted and compared against a threshold.

```
protected synchronized boolean
   interpretInform(Message currentMessage) {
      if (currentMessage.getContent()
         instanceof Plumage) {
            Plumage courterPlumage =
               (Plumage) currentMessage.getContent();
            strategy.getLearningModule()
               addInformation(courterPlumage);
```

**The learning algorithm of Smith and Taylor (1998)** If an agent has received enough plumage items, he chooses the genotype with the highest fitness and starts the learning process performing crossover and mutation. The new genotype substitutes the old genotype. The learning is only started, if there are enough choices in form of plumage items.

```
if (strategy.getLearningModule().
   getRecommendedAction() ==
      ILearning.MATE) {
      strategy.setMyGenotype(
         (Genotype) strategy.getLearningModule().
            learn(strategy.getMyGenotype()));
```

The parameter maturityThreshold is checked which prevents a too fast changing of the strategy. This equals the sexual maturity in biology. If an agent is old enough, he is allowed to distribute his genes with his genotype.

```
Smith98.learn() public Object learn(Object oldGenotype) {
   if (maturity < maturityThreshold) {
      return oldGenotype;
      }
   Plumage p = getMate();
   if (p != null) {
      Genotype g = learningOwner.getStrategy().
         getMyGenotype().cross(p.getSenderGenes(),
         new java.util.Random());
      g.mutate(new java.util.Random());
      return g;
      }
   // no partner found, repeat old genotype
   synchronized (courterPlumages) {
```

```
        courterPlumages.clear();
      }
   return oldGenotype;
   }
```

In the next step, the plumage with the highest fitness is selected. After selection of a learning partner, both genotypes are recombined to a new genotype:

```
//crossover
public Genotype cross(Genotype geneOfFather,
   Random Rand) {
      FloatGene g;
      Genotype ret = new Genotype();
      if (size() != geneOfFather.size()) {
      // genes of father and mother
      // have different length");
      }
      Enumeration ep = elements();
      Enumeration em = geneOfFather.elements();
      while (ep.hasMoreElements()) {
         FloatGene gp = (FloatGene) ep.nextElement();
         FloatGene gm = (FloatGene) em.nextElement();
         if (Rand.nextFloat() < 0.5) {
            ret.add(
               new FloatGene(gp.getDoubleValue()));
         } else {
            ret.add(
               new FloatGene(gm.getDoubleValue()));
         }
      }
      return ret;
   }
```

The method cross() adds with a percentage of 50 either the mother or father gene to the new genotype. The resulting child genotype has a combination of mother and father genes at the end. The recombination phase is followed by the mutation phase. This phase performs some changes at the child genotype (mutate()). This method closes the adaptation. A garbage collection process resets the plumage-lists.

## 3.4 Relations to other Work Packages

The conceptualization and definition of the centralized and decentralized market mechanisms for the service market in this chapter relate strongly to the efforts in work package 2 (simulator), work package 3 (proof of concept) and work package 4 (evaluation).

Both, the simulator and the prototype have to support the provisioning of standardized services. This must be carried out in a way, that sequences of basic services may be requested. A common interface for the specification of services in the simulator as well as the prototype would be very helpful. In the following these relations are elaborated in more detail.

### 3.4.1 Relations to Simulator (WP2)

In the simulator work package (WP2) the main focus within the first 12 months of the project duration was the selection of the appropriate simulation tool. The main desiderata for the selection criteria is the compatibility of the simulation tool with the desired market mechanisms.

On the one hand, the centralized service auctioneers and on the other hand, decentralized – catallactic – market mechanisms must be covered by simulations. Therefore, the centralized and the decentralized mechanisms developed throughout this chapter are relevant to the selection of the appropriate simulator tool.

In the second year of the project, the interconnection between work package 1 and 2 will be even more emphasized than in the first year, when the developed market mechanisms will be implemented in the simulator.

### 3.4.2 Relations to Proof of Concept (WP3)

Based on existing P2P architectures, the contribution of the outcomes of work package 1 to work package 3 is the definition of the relation between catallactic market mechanisms and Grid/P2P systems. Here, the concepts of decentralized allocation mechanisms will be taken into account when defining and implementing the prototype setup.

### 3.4.3 Relations to Evaluation (WP4)

There exists a fundamental relation between work package 1 and work package 4. The main role of work package 4 in the project is to define and identify relevant metrics for the measurement of economic and technical efficiency of the market mechanisms.

These metrics will then be applied in order to evaluate the efficiency of the developed centralized market mechanisms versus the quality of the decentralized – catallactic – market mechanisms with respect to allocative efficiency, incentive efficiency, budget balance, rationality, tractability, communication complexity, and technical efficiency. One of the key goals will be to apply the efficiency measures to the data, obtained from simulations and the prototype application.

# Chapter 4

# Specifications of the Resource Market

## 4.1 Environmental Analysis

Having defined the environment for trading the services, the corresponding market for trading the resources has to be designed. The environment for designing a resource market can be compared to the environments found in the Grid, Peer-to-Peer, and Distributed Computing literature. These research areas – especially the research done in the area of Computational Grids – build the base for the following analysis.

### 4.1.1 Environment Definition

The environment of the resource market mainly comprises basic services and resource services:

**Basic Service** A basic service is a modular software application which needs a set of resources for fulfilling its goals, i.e. for executing a specific application, that a complex service requires.

**Resource Service** A resource service is a computing resource which encapsulates the computation capabilities as a service.

Figure 4.1 outlines the resource market of the CATNETS scenario. Suppose a basic service is part of the allocation in the service market with a PDF service. The basic service now requires computer resources for executing the service (e.g. a computation service capable providing 400 MIPS and 512 MB of memory and a storage service with 10 GB of free space). Thus, the basic service acts as a resource consumer on the resource market and the so-called resource services as resource suppliers.
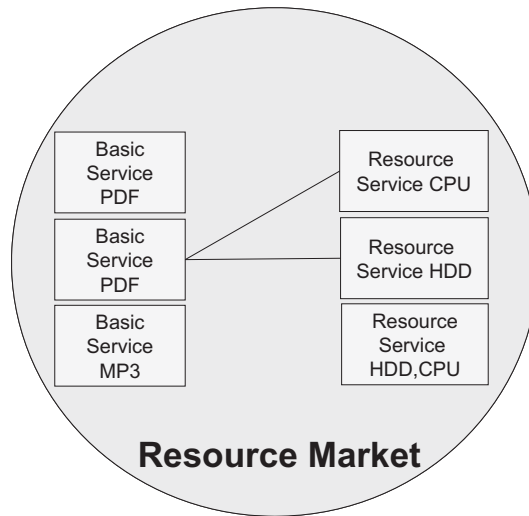
Figure 4.1: Resource Market

**Participants**

The resource market – in analogy to Grid environments – consists of multiple buyers and sellers. Basic services act as buyers and resource services as sellers.

In analogy to the service market, it is difficult to determine a realistic number of participants in a market for trading computer resource. At the moment, there is no existing commercial market place for trading such resources. However, there exist several test-beds and experimental evaluation studies which serve as clues. Table 4.1 lists exemplarily a selection of test-beds found in the Grid, Peer-to-Peer, and Distributed Computing literature.

| Name | Resources | Users | Jobs/Day |
|---|---|---|---|
| Grid 2003 Project | 700 | 102 | 1300 |
| NorduGrid | | | 1300 |
| Conrell Center | 512 | 512 | 800 |
| San Diego | 1152 | 468 | 250 |
| PlanetLab | 350 | 350 | |

Table 4.1: Statistics of Grid and Peer-to-Peer test-beds.

**Grid 2003 Project:** The Grid2003 Project deploys a multi-virtual organization and application-driven Grid laboratory. The project sustained the production-level services for several months which were required by several different research institutes (e.g. for physics experiment, gravitational wave search) [FGG+04]. In peak times, the system handled 1300 jobs simultaneously.

**NorduGrid:** The NorduGrid[1] project provides a Grid infrastructure by adapting existing Grid middleware components (e.g. Globus Toolkit). In peak times, the NorduGrid handles about 1300 jobs per day.

**Cornell Theory Center:** The Cornell Theory Center is a high-performance computing center which supports scientific and engineering research projects. The center has over 500 CPUs and handles over 800 jobs per day[2] in peak times.

**San Diego Supercomputer:** The San Diego Supercomputer Center hosts a cluster with over 1100 processors. The average number of jobs executed per day was measured with 250 [EHY02]. The workload of this center served also as a benchmark for several scheduling architectures, e.g. the workload which is used in [EHY02].

**PlanetLab:** PlanetLab is a distributed overlay platform to support the deployment and evaluation of planetary-scale network services [BBC$^+$04b]. It includes over 350 machines located in over 20 different countries.

Besides several test-beds, there exist simulation settings for evaluating the performance of different Grid components (e.g. scheduler, resource manager). For instance, [IF01] use in their setting 10.000 resources, 5.000 different users, and 100 jobs per day. [KDJN03] perform an evaluation with 500 different users. [RS04] use 6.000 different resources for their simulation setting.

Similar to the analysis for the environment for the service market, the related systems and simulation settings give a first impression of the number of participants and the number of jobs to model. Moreover, the number of participants in the resource market strongly depends on the overall number of participants (i.e. complex services and basic services) in the service market. Therefore, a detailed analysis has to be made in conjunction with the service market environment as well as with the technical analysts in the CATNETS project.

**Transaction Objects**

In the resource market, the transaction objects are computing resources like processors or storage servers. Computing resources are non standardized capacity-type objects. Capacity (e.g. processing power) not used in a time-period $t$ is worthless at a later time-period $t + 1$. Furthermore, the same resources (e.g. CPUs) can differ in their capacities, e.g. a computing engine can be a high-performance processor capable of processing billions of

---

[1]See `http://www.nordugrid.org/` and `http://grid.uio.no/atlas/nglogger/nglogger_info/` for details (accessed: 10.02.2005).

[2]See `http://www.cs.huji.ac.il/labs/parallel/workload/` for details (accessed 05.01.2005)

MIPS[3], as well as a low-performance processor of a mobile hand-held computer. There-fore, computer resources can be described by a set of possible attributes and its character-istics. For instance, a storage server can be characterized by its quality attributes capacity (in Gigabyte (GB)), access time (in milliseconds (ms)), and data throughput (in bits per second (bits/s)).

The set and the number of possible different transaction objects in the resource market (e.g. resources like CPUs, hard-disks) are not specified in detail. Surveying the literature, a common set of possible resources can be identified. [CS01] introduce a base set of re-source types that have to be supported within a resource management mechanism: Com-puter resources (CPU, GPU), storage resources (memory, disk, tape), network resources (HPC switch, LAN, WAN), and miscellaneous resources (guest account, quota protocol). [BM02] and [KC02] abstract to the following five resources: processor, memory, storage, I/O, and network.

The specification of Open Service Grid Architecture (OGSA) as an open standard for the interactions between different computing resources across organizational entities goes one step further: OGSA defines computer and storage resources as well as networks, pro-grams, databases, and the like as services, i.e. network enabled entities that provide some capability. Conceiving any use of resources as service paves the path for interoperabil-ity among heterogeneous computing and application environments [FNT02]. OGSA and one of its reference implementations Globus Toolkit[4] provide the technical infrastructure for accessing computational services over the Grid. Computers equipped with a Globus installation can easily access computational resources over the Grid, where the access is in-transparent for the user. OGSA considers services which reflect resource function-alities (e.g. storage) and quality characteristics (e.g. size), dependencies, and time at-tributes. Relying on services instead of computational resources removes many technical problems. For instance, the resource CPU may technically not be offered without some amount of hard disk on the same computer, while a computation service offering CPU cy-cles already includes the complementary resources. In the remainder of the deliverable, it is abstracted from those technical details by treating resources and services as synonyms.

Concluding the discussion, the transaction objects are seen as services with multiple attributes. For the concrete number of different services, the reader is referred to deliver-able D2.1 (Simulation) [Zin05].

## 4.1.2   Requirement Analysis

Despite the classical mechanism design requirements pertaining to the mechanism as de-scribed in section 3.1.2, the mechanism for the resource market must also account for the underlying environment. For the resource market the requirements stemming from the

---

[3]MIPS stands for million instructions per second and is an approximate measure of a computer's pro-cessing speed.

[4]See `http://www.globus.org/` for details.

underlying environment are the following [SNW04, SNVW05a]:

**Simultaneous trading and immediate resource allocation:** In analogy to the service market requirements, the market mechanism for the resource market has to support the simultaneous trading of multiple buyers and sellers, as well as an immediate resource allocation.

**Trading dependent resources:** In the resource market, buyers usually demand a combination of computer resources as a bundle to perform a task [SMT02]. This bases on the fact that computer resources (e.g. in the Computational Grid) are complementarities. Complementarities are resources with superadditive valuations $(v(A) + v(B) \leq v(AB))$, as the sum of the valuations for the single resources is less than the valuation for the whole bundle.

Suppose, for example, a buyer intending to render images requires hard disk, CPU, and main memory. If any component of the bundle, say the CPU, is not allocated to him, the remaining bundle has no value for him since the rendering cannot be processed without the CPU. In order to avoid the exposure risk (i.e. receiving only one part of the bundle[5] without the other), the mechanism must allow for bids on bundles. Likewise, the seller can also express bids on bundles.

**Support for multi-attribute resources:** Resources in the resource market are typically not completely standardized, as similar resources can differ in their quality. A hard disk can be characterized by its quality attributes capacity (in Gigabyte (GB)), access time (in milliseconds (ms)), and data throughput (in bits per second (bits/s)). For example, a rendering job requiring a minimum amount of 250 GB can be conducted by a 500 GB hard disk; however, it can not be executed by a 100 GB hard disk. As such, minimum quality requirements must be met, while similar resources of superior quality work as well. Hence, minimum quality requirements must be met, while similar resources of superior quality work as well[6].

**Language includes co-allocation constraints:** Capacity demanding Grid applications usually require allocating simultaneously several homogenous service instances from different providers. For instance, a large scaled simulation may require several computation services to be completed in time. In the Grid literature, the simultaneous allocation of multiple homogenous services is often referred to as co-allocation [AMA04, CFK04]. A mechanism for the Grid has to enable co-allocations and has to provide functionality to control this: Firstly, it is desirable to limit the maximal number of co-allocations for services, i.e. to limit the maximum number of divisions of a service. For instance, a buyer wants to execute a specific application that requires a computation service which can be allocated at most from three different

---

[5]A bundle denotes a combination of resources.

[6]The network congestion can also be included by the use of attributes. Therefore, network congestion metrics have to be defined and evaluated in future work, e.g. [Wol98], [Loe04]

sellers. Secondly, it is required to couple multiple services of a bundle, i.e. to guarantee that these resources are allocated from the same seller and – as such – will be executed on the same machine. Suppose a legacy application requires storage and computation. Due to application restrictions it can only be executed on a single machine.

**No partial execution on the buyers' side:** The mechanism is required to avoid partial executions of services. Any partial execution of a buyer's request is useless.

**Partial execution on the sellers' side:** A seller of resources (e.g. an owner of a scaled server farm) is also willing to sell just a part of the offered resources. Thus, the possibility of a partial executing on the sellers' side must be given.

## 4.2   Meeting the Requirements – Related Work

A number of mechanisms have been proposed that attempt to solve the resource allocation problem in such a resource market or related architectures. Most of these mechanisms are central in nature in a way that the allocation problem is solved by a central entity using global optimization algorithms without the employment of prices. This central entity requires detailed information about the demand and supply situation in order to be effective. As information is dispersed among the buyers and sellers, central allocation algorithms may not enfold their full power, because this information requirement is not even closely met. Market-based approaches incorporate incentives for truthful information revelation by implementing prices.

Surveying the literature, various mechanisms for allocating computer resources can be found. Most approaches adapt classical auction mechanism or decentralized bargaining schemas for standardized products as described in section 3.2. Furthermore, there exists a number of multi-attribute, bundle, or combinatorial mechanisms, which can be used to trade dependent and non-standardized products. Both types of mechanisms and their suitability for the resource market will be discussed in the following.

### 4.2.1   Centralized Mechanisms

The use of market mechanisms for allocating computer resources is not a completely new phenomenon[7]. [WHH+92] and [RN98] propose the application of Vickrey auctions for the allocation of homogenous computational resources in distributed systems. Vickrey auctions achieve truthful bidding as a dominant strategy and hence result in efficient allocations.

---

[7]A detailed overview of classical auctions in resource allocation systems is given among others in [KBM02].

[BSGA01] were among the first who motivated the transfer of market-based systems from distributed systems to Grids. Nonetheless, they propose classical one-sided auction types, which cannot account for combinatorial bids. [WPBB01b] compare classical auctions with a bargaining market. As a result, they come to the conclusion that the bargaining market is superior to an auction based market. This result is less surprising, as the authors only consider classical auction formats, where buyers cannot express bids on bundles.

[SMT02] account for combinatorial bids by providing a tâtonnement process for allocation and pricing Grid resources. Furthermore, [NBC$^+$05] propose repeated combinatorial auctions as a microeconomic resource allocator in distributed systems. Nonetheless, the resources are still considered to be standardized commodities. Standardization of the resources would either imply that the number of resources are limited compared to the number of all possible resources or that there are many mechanisms, which are likely to suffer under fewer participation. Both implications result in rather inefficient allocations.

Additionally, state-of-the-art mechanisms widely neglect co-allocation attributes for bundles and quality constraints for single resources. Hence, the use of these mechanisms in the Grid environment is considerably diminished.

[WWWMM01] model single-sided auction protocols for the allocation and scheduling of resources under consideration of different time constraints. [Con02] goes one step further by designing a combinatorial bidding procedure for job scheduling including different running, starting, and ending times of jobs on a processing machine. However, these approaches are single-sided and favour monopolistic sellers or monopsonistic buyers in a way that they allocate greater portions of the surplus. Installing competition on both sides is deemed superior, as no particular market side is systematically put at advantage.

Demanding competition on both sides suggests the development of a combinatorial exchange. In literature, [PKE01] introduce the first combinatorial exchange as a single-shot sealed bid auction. As payment scheme, Vickrey discounts are approximated. [BN04] and [PCE$^+$05] propose an iterative combinatorial exchanges for the allocation problem. By doing so, the preference elicitation problem can be alleviated, as the bidders can restrict their attention to some preferred bundles in contrast to all $2^G - 1$ possible combinations, where $G$ is the number of different resources. Obviously, these approaches do not account for quality constraints and are thus not directly applicable for the Grid allocation problem.

Counteractively, [BDGS05] propose a family of combinatorial auctions for allocating Grid services. Although, the mechanism accounts for quality and time attributes and enables the simultaneous trading of multiple buyers and sellers, there is no competition on the sellers' side as these orders are aggregated to one virtual order. Moreover, the mechanism neglects co-allocation constraints.

Table 4.2 summarizes the aforementioned results and reviews the related mechanisms according to the requirements presented in section 4.1.2. The participants row denotes,

| Auction | Participants | Direction | Combinatorial Bids | Quality Attributes | Co-Allocation |
|---|---|---|---|---|---|
| [RN98], [WHH$^+$92] | $1:n$ | sealed-bid | no | no | no |
| [BSGA01] | $1:n$ | iterative | no | no | no |
| [WPBB01b] | $1:1$ | iterative | no | no | no |
| [SMT02] | $1:1$ | iterative | yes | no | no |
| [WWWMM01] | $1:n$ | iterative | yes | no | no |
| [Con02] | $1:n$ | iterative | yes | no | no |
| [NBC$^+$05] | $m \bullet (1:n)$ | sealed-bid | yes | no | no |
| [PKE01] | $m:n$ | sealed-bid | yes | no | no |
| [BN04], [PCE$^+$05] | $m:n$ | iterative | yes | no | no |
| [BDGS05] | $m \bullet (1:n)$ | sealed-bid | yes | no | yes |

Table 4.2: Market Mechanisms for Grid

whether a mechanism is single-sided $(1:n)$ or double-sided $(m:n)$. Multiple or aggregated instances of single-sided mechanisms are denoted by $m\bullet(1:n)$. The directions row indicates if the mechanism is a one-shot (sealed-bid) or a multiple round game (iterative). The further rows classify the mechanisms concerning their support for combinatorial bids, quality attributes, and co-allocation restrictions.

As table 4.2 demonstrates, there is no market mechanism that installs competition on both sides, includes combinatorial bids, allows for time constraints, manages quality constraints, and considers co-allocation constraints.

Furthermore, there exists a number of approaches applying mechanisms – mostly auctions – to Grid related domains like the energy market or machine scheduling tasks. As these domains evince differences to the proposed Grid scenario, the related mechanisms are not appropriable.

Reviewing the requirements for a resource market, it becomes obviously that none of the presented classical auction types and combinatorial mechanisms is directly applicable for the central resource market. None of the mechanisms comprise at the same time simultaneous bids of multiple sellers and buyers, quality and co-allocation constraints, and combinatorial bids.

This deliverable intends to fill this need by presenting the design of a multi-attribute combinatorial exchange for allocating and scheduling Grid resources as described in [SNW04] and [SNVW05a].

### 4.2.2 Decentralized Mechanisms

The basic concepts for the decentralized mechanism for the resource market are the same as on the service market. Therefore, the related work in this section concentrates on bundling. Bundling influences the decentralized mechanism in the search for bundles and their creation. Bundles group different resource providers together delivering the demanded resources. It is expected that the communication and computing complexity will increase using the bundling concept.

Social welfare economic mainly concentrates on the communication complexity of the negotiation in multi-agent systems. They bring together ideas from the socioeconomic sciences on one hand, and from computing and AI on the other hand [ASS02] [DWL03]. The EU project Societies of ComputeeS (SOCS, IST-2001-32530) designs simulations of societies for distributed resource allocation developing negotiation heuristics [Soc]. The agents (computees) have incomplete knowledge and have to coordinate and cooperate with other agents to achieve local or global objectives.

Besides the communication complexity, there is also a need for a fast heuristic strategy creating bundles. In economics, similar problems arise in planning and production control. The system status in a production environment is continuously changing and has to be adapted very fast for an efficient usage of the machinery. Research on using fuzzy logic systems enables a dynamic scheduling of machinery in a production plant [Keu99]. In CATNETS, we have standardized basic services, which deliver a certain quality of service. Different scheduling strategies for different quality of service can be used to create resource bundles. If a fast allocation of resources is the objective of a "gold"-QoS basic service, a first come first serve strategy should used for bundle creation.

### 4.2.3 Summary

Summarizing the related work it is obvious, that no existing market mechanisms fits all requirements defined in section 4.1.2. As such, a multi-attribute combinatorial exchange is designed and implemented within the CATNETS project. The decentralized market mechanism uses the concepts presented at the service market enhancing them with the bundling requirements of the resource market.

## 4.3 Design of the Resource Market

This section will provide a detailed presentation of the market models on the resource market. In analogy to the service market in chapter 3, basic design issues like bidding language and message specification, which are the same on both mechanisms, are defined in subsection 4.3.1. The centralized market mechanisms are specified in subsection 4.3.2 and section 4.3.3 presents the decentralized market.

### 4.3.1 Bidding Language and Message Specification

Allowing participants to submit multi-attribute combinatorial bids requires a formalized bidding language that meets the above mentioned requirements:[8]

Let $\mathcal{N}$ be a set of $N$ buyers and $\mathcal{M}$ be a set of $M$ sellers, where $n \in \mathcal{N}$ defines an arbitrary buyer and $m \in \mathcal{M}$ an arbitrary seller. Furthermore, there is a set of $G$ discrete resources $\mathcal{G} = \{g_1, \ldots, g_G\}$ with $g_k \in \mathcal{G}$ and a set of $S$ bundles $\mathcal{S} = \{S_1, \ldots, S_S\}$ with $S_i \in \mathcal{S}$ and $S_i \subseteq \mathcal{G}$ as a subset of resources. For instance, the bundle $S_i = \{g_j, g_k\}$ denotes that the bundle $S_i$ consists of the two resources $g_j$ and $g_k$.

A resource $g_k$ has a set of $A_k$ cardinal quality attributes $\mathcal{A}_{g_k} = \{a_{g_k,1}, \ldots, a_{g_k,A_k}\}$ where $a_{g_k,j} \in \mathcal{A}_{g_k}$ represents the $j.th$ attribute of the resource $g_k$.

A buyer $n$ can specify the minimal required quality characteristics for a bundle $S_i \in \mathcal{S}$ with $q_n(S_i, g_k, a_{g_k,j}) \geq 0$, where $g_k \in S_i$ is a resource of the bundle $S_i$ and $a_{g_k,j} \in \mathcal{A}_{g_k}$ is a corresponding attribute of the resource $g_k$. Accordingly, a seller $m$ can specify the maximal offered quality characteristics with $q_m(S_i, g_k, a_{g_k,j}) \geq 0$. For instance, in the context of resource services, a quality attribute can be the $size$ of a storage service with a possible corresponding characteristic of $200GB$.

For each resource $g_k \in S_i$, a buyer $n$ can specify the maximum number of co-allocations with $\gamma_n(S_i, g_k) \geq 0$. This means, that a buyer $n$ can limit the number of partial executions for each resource $g_k$. For instance, the buyer may want to ensure that a computation job is been split at most on 4 different machines. Let $\gamma_n(S_i, g_k) = D$, if the resource $g_k$ has no divisibility restrictions, where $D$ is a large enough constant.[9] The coupling of two resources in a bundle is represented by $\varphi_n(S_i, g_i, g_j) \in \{0, 1\}$ where $\varphi_n(S_i, g_i, g_j) = 1$ if the resources $g_i$ and $g_j$ have to be allocated from the same seller and $\varphi_n(S_i, g_i, g_j) = 0$ otherwise. The coupling restriction is helpful if a buyer may want to ensure that, for instance, a computation service as well as a corresponding storage service for a specific job are executed on the same machine.

A buyer $n$ can express the valuation for a bundle $S_i$ by $v_n(S_i) \geq 0$, i.e. the maximum price for which the buyer $n$ is willing to trade. The reservation price for allocating a bundle $S_i$ is denoted by $r_m(S_i) \geq 0$, which represents the minimum price for which the seller $m$ is willing to trade.

Subsequently, a buyer $n$ can submit a bid $B_n(S_i)$ defined as the tuple

$$
\begin{aligned}
B_n(S_i) = &(v_n(S_i), q_n(S_i, g_1, a_{g_1,1}), \ldots, q_n(S_i, g_G, a_{g_G,A_j}), \\
&\gamma_n(S_i, g_1), \ldots, \gamma_n(S_i, g_G), \\
&\varphi_n(S_i, g_1, g_2), \varphi_n(S_i, g_1, g_3), \ldots, \varphi_n(S_i, g_1, g_G), \ldots, \varphi_n(S_i, g_G, g_{G-1}).
\end{aligned}
$$

---

[8]The bidding language neglects time restrictions, i.e. attributes concerning the duration, start, and end time of a job. For a detailed description on how these attributes can be included, the reader is referred to [SNVW05a, SNVW05b].

[9]The constant $D$ has to be greater than the total number of seller bids.

For example, the bid

$$B_n(S_1) = \{10, ((32, 200), 1000), (0, 4), (0)\} \qquad (4.1)$$

with $S_1 = \{g_1, g_2\}$, $g_1 = \{Computation\}$ with two attribute $\mathcal{A}_{g_1,1} = \{numberOfCPUs\}$ and $\mathcal{A}_{g_1,2} = \{memoryPerCPU\}$, and $g_2 = \{Storage\}$ with one attribute $\mathcal{A}_{g_2} = \{storageSpace\}$ expresses that a buyer $n$ wants to buy the bundle $S_1$ and has a valuation of $v_n(S_1) = 10$ for it. The requested computation service $g_1$ should have at least 32 CPUs and each CPU at lest 200 MB of main memory; the storage service $g_2$ should have at least 1000 GB available space. The computation service $g_1$ can be at most split on four different machines simultaneously. Furthermore, the resources have no coupling restrictions ($\varphi_n(S_1, g_1, g_2) = 0$).

The orders of the sellers are formalized in a similar way as the buyers' orders are. They do, however, not include maximum divisibility and coupling properties. An order $B_m(S_i)$ of a seller $m$ for a bundle $S_i$ is defined as the tuple

$$B_m(S_i) = (r_m(S_i), (q_m(S_i, g_1, a_{g_1,1}), \ldots, q_m(S_i, g_j, a_{g_j,A_j}))).$$

The bidding language can be mapped to most parts of common agreement specifications defined by the Grid community, e.g. WS-Agreement [ACD+05]. The mapping from an existing WS-Agreement to the bidding language is also possible. The following XML encoded document depicts a WS-Agreement encoded offer for the above described bidding example (4.1):

```
<wsag:AgreementOffer>
 <wsag:Terms>
  <wsag:All>
    <wsag:ServiceDescriptionTerm wsag:Name="executable"
      wsag:ServiceName="ServerService">
      <job:executable>/catnets/file</job:executable>
    </wsag:ServiceDescriptionTerm>
    <wsag:ServiceDescriptionTerm wsag:Name="arguments"
      wsag:ServiceName="ServerService">
      <job:arguments>/some/argument</job:arguments>
     </wsag:ServiceDescriptionTerm>
    <wsag:ServiceDescriptionTerm wsag:Name="readFile"
      wsag:ServiceName="StorageService">
     <job:arguments>/some/file/to/read</job:arguments>
    </wsag:ServiceDescriptionTerm>
    <wsag:All>
     <wsag:ServiceDescriptionTerm wsag:Name="numberOfCPUs"
       wsag:ServiceName="ServerService">
       <job:numberOfCPUs>32</job:numberOfCPUs>
```

```
      </wsag:ServiceDescriptionTerm>
      <wsag:ServiceDescriptionTerm wsag:Name="memoryPerCPU"
        wsag:ServiceName="ServerService">
        <job:realMemorySize>200</job:realMemorySize>
      </wsag:ServiceDescriptionTerm>
      <wsag:ServiceDescriptionTerm wsag:Name="storageSpace"
        wsag:ServiceName="StorageService">
        <job:realMemorySize>1000</job:realMemorySize>
      </wsag:ServiceDescriptionTerm>
     </wsag:All>
     <wsag:GuaranteeTerm wsag:Name="Valuation">
      <wsag:ServiceScope>
       <wsag:ServiceName>ServerService</wsag:ServiceName>
        <wsag:ServiceName>StorageService</wsag:ServiceName>
       </wsag:ServiceScope>
       <wsag:BusinessValueList>
        <wsag:CustomBusinessValue>
         <catnets:reservation>10</catnets:reservation>
        </wsag:CustomBusinessValue>
       </wsag:BusinessValueList>
      </wsag:GuaranteeTerm>
      <wsag:GuaranteeTerm wsag:Name="Coupling">
       <wsag:ServiceScope>
        <wsag:ServiceName>ServerService</wsag:ServiceName>
        <wsag:ServiceName>StorageService</wsag:ServiceName>
       </wsag:ServiceScope>
       <wsag:QualifyingCondition>
        <catnets:coupling>true</catnets:coupling>
       </wsag:QualifyingCondition>
      </wsag:GuaranteeTerm>
      <wsag:GuaranteeTerm wsag:Name="MaxDivisibility">
       <wsag:ServiceScope>
        <wsag:ServiceName>ServerService</wsag:ServiceName>
       </wsag:ServiceScope>
       <wsag:QualifyingCondition>
         <catnets:maxDivisibility>4</catnets:maxDivisibility>
       </wsag:QualifyingCondition>
      </wsag:GuaranteeTerm>
     </wsag:All>
   </wsag:Terms>
</wsag:AgreementOffer>
```

The document bases on WS-Agreement and JSDL (Job Specification Description Language) specifications. The co-allocation constraints (divisibility and splitting) as well as the valuation and reservation prices were added as domain specific schemas using a separate XML-schema (cf. section 3.3.1).

In the following, a central and decentral mechanism for coordinating the resource market are designed on base of this bidding language.

## 4.3.2 Centralized Market

As shown in the environmental analysis (cf. section 4.1), services can be characterized as multi-attribute heterogeneous commodities. In analogy to the service market, an auction schema is chosen, as auctions can allocate resources efficiently. Complying with the requirements defined in section 4.1.2, a multi-attribute combinatorial double auction is designed.

Firstly, the integration of a central market mechanism into existing Grid architectures is discussed. After that, the schema is introduced by the definition of a winner determination rule and a pricing schema.

**Resource Market**

Following the above definitions of the environment and surveying the Grid literature, the simplified picture in Figure 4.2 can be adapted as a market place for the resource market [CFK04]. In essence, the market for trading computer resources (e.g. for trading Grid resources) is spanned around the resource owners as sellers (e.g. IBM or Sun with their computer centers), the resource consumers as buyers (e.g. scientists at universities, rendering or the biochemical firms) and some intermediaries (e.g. Condor, Gallop, Legion etc.). The intermediaries technically provide the resource management infrastructure for exploiting remote resources.
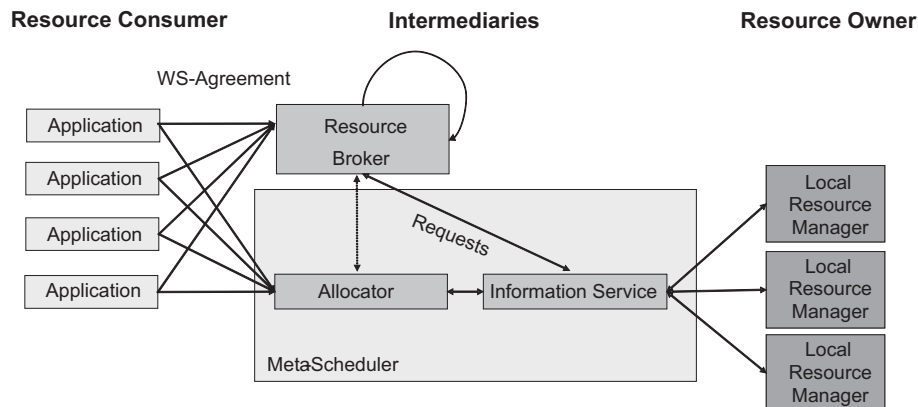


Figure 4.2: The market for Computational Grids

According to the resource management architecture proposed by [CFK04] the intermediary layer consists of three basic components:

**Resource Broker:** The resource broker components are responsible for resource discovery, selection, aggregation, and subsequently for the data and program transportation [CB02]. By transforming the resources to the consumers' requirements (which are specified for instance in WS-Agreement) into a set of jobs that are self-reliantly

scheduled on the appropriate resources and subsequently managed, the complexity of the resource market is concealed [VB04]. For the market participants, the resource broker is apparently more of a black box. A resource broker may also include a matchmaking between multi-attribute offers and request [Vei03].

**Resource Information Manager:** The resource information manager provides pervasive access to information about the current availability and capability of resources [CFK04].

**Allocator:** The allocator coordinates the allocation of resources at multiple sites. Obviously, the allocator and the information service assume the responsibility of (meta-) scheduling the jobs.

Based upon this view on the intermediary layer, the market mechanism for the resource market can be – in analogy to a market mechanism for the Computational Grid – sketched as follows: The transition from an intermediary layer to a mediated market mechanism is not too far. In essence, the scheduling performed by the resource broker can be shifted to the market mechanism. Instead of sending requests to the information service, the resource broker can translate the user requirements into bids. Those bids expressing demand and supply situation are subsequently cleared by the market mechanisms [SNW04].

Figure 4.1 depicts the high level architecture of the resource market for CATNETS. Basic services can submit sell orders to the order books, e.g. formulated as a WS-Agreement encoded offer. In the same way, resource services can submit buy orders to the order books. Instead of having multiple order books as in the service market, one central order book is used to guarantee the allocation of several homogeneous goods as a bundle.

After the participants submitted their bids to the auctioneer, the allocation (winner determination) and the corresponding prices are determined.

**Winner Determination**

Based upon the formalized bidding language defined in section 4.3.1, the multi-attribute combinatorial winner determination problem (WDP) with the objective of achieving allocative efficiency can be formalized as a linear mixed integer programm as an extension of the model presented in [SNW04, SNVW05a, GSVW05]:

Let $x_n(S_i) \in \{0, 1\}$, $y_{m,n}(S_i) \geq 0$, and $d_{m,n}(S_i) \in \{0, 1\}$ be the decision variables of the programm. The binary variable $x_n(S_i)$ denotes, whether the bundle $S_i$ is allocated to the buyer $n$ ($x_n(S_i) = 1$) or not ($x_n(S_i) = 0$). For a seller $m$, the real-valued variable $y_{m,n}(S_i)$ with $0 \leq y_{m,n}(S_i) \leq 1$ indicates the percentage contingent of the bundle $S_i$ allocated to the buyer $n$. For example, $y_{m,n}(S_i) = 0.5$ denotes that 50 percent of the qualities of the bundle $S_i$ are allocated from seller $m$ to buyer $n$. In the previous example
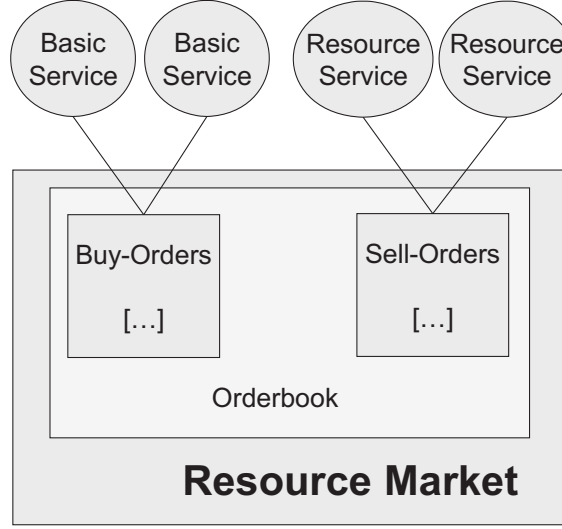
Figure 4.3: The market for Computational Grids

including the 1000 GB storage service, a partial allocation of 500 GB from seller $m$ to buyer $n$ would lead to $y_{m,n}(StorageService) = 0.5$.

Furthermore, the binary variable $d_{m,n}(S_i)$ is concatenated with $y_{m,n}(S_i)$ and denotes whether the seller $m$ allocates to the buyer $n$ $d_{m,n}(S_i) = 1$ or not $d_{m,n}(S_i) = 0$.

$$\max \sum_{n \in \mathcal{N}} \sum_{S_i \in \mathcal{S}} v_n(S_i) x_n(S_i) - \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} \sum_{S_i \in \mathcal{S}} r_m(S_i) y_{m,n}(S_i) \tag{4.2}$$

$$\sum_{n \in \mathcal{N}} y_{m,n}(S_i) \leq 1, \forall m \in \mathcal{M}, \forall S_i \in \mathcal{S} \tag{4.3}$$

$$\sum_{S_i \ni g_k} x_n(S_i) q_n(S_i, g_k, a_{g_k,j}) - \sum_{S_i \ni g_k} \sum_{m \in \mathcal{M}} y_{m,n}(S_i) q_m(S_i, g_k, a_{g_k,j}) \leq 0 \,,$$
$$\forall n \in \mathcal{N}, \forall g_k \in \mathcal{G}, \forall a_{g_k,j} \in \mathcal{A}_{g_k} \tag{4.4}$$

$$\sum_{S_i \ni g_k} \sum_{m \in \mathcal{M}} d_{m,n}(S_i) - \sum_{S_i \ni g_k} \gamma_n(S_i, g_k) x_n(S_i) \leq 0,$$
$$\forall n \in \mathcal{N}, \forall g_k \in \mathcal{G} \tag{4.5}$$

$$\sum_{S_j \ni g_k, g_l} \varphi_n(S_j, g_k, g_l) \left( \sum_{S_i \ni g_k} d_{m,n}(S_i) - \sum_{S_i \ni g_l} d_{m,n}(S_i) \right) = 0,$$
$$\forall n \in \mathcal{N}, \forall m \in \mathcal{M}, \forall g_k, g_l \in \mathcal{G} \tag{4.6}$$

$$\sum_{S_j \ni g_k, g_l} \varphi_n(S_j, g_k, g_l)(\sum_{S_i \ni g_i} \sum_{m \in \mathcal{M}} d_{m,n}(S_i) + \sum_{S_i \ni g_j} \sum_{m \in \mathcal{M}} d_{m,n}(S_i) - 2x_n(S_j)) \le 0,$$

$$\forall n \in \mathcal{N}, \forall g_k, g_l \in \mathcal{G} \tag{4.7}$$

The objective function (4.2) maximizes the surplus $V^*$ which is defined as the difference between the sum of the buyer's valuations $v_n(S_i)$ and the sum of the sellers' reservation prices $r_m(S_i)$. Assuming truthful bidders, the objective function reflects the goal of maximizing the social welfare in the economy. Constraint (4.3) guarantees that each seller cannot allocate more than the seller possesses.

Constraint (4.4) ensures that for any allocated bundle, all required resources have to be fulfilled in at least the demanded qualities. Constraint (4.5) ensures that a resource will be provided by at most $\gamma_n(S_i, g_k)$ different suppliers . The constraints (4.6) and (4.7) account for the coupling of two resources. Constraint (4.6) ensures that two resources have to be provided by the same seller, in case they should be coupled. This constraint alone does not suffice the coupling requirements of resources. It could happen, that a coupled computation service with 3000 MIPS and a storage service with 30 GB would be co-allocated by two sellers in different qualities, e.g. a computation service with 2999 MIPS and a storage service with 1 GB from one seller and a computation service with 1 MIPS and a storage service with 29 GB from another one. To exclude those undesirable allocations, constraint (4.7) impose the restriction that coupled resources can not be co-allocated. Simplifying the model, this also includes free-disposal resources. For instance, if the computation service with 3000 MIPS and the storage service with 30 GB are allocated from one particular seller as a bundle, another seller cannot allocate a bundle containing a rendering service and another storage service to the same buyer. However, the seller may allocate any bundle without a storage and computation service to the buyer, e.g. the rendering service alone.

Finally, an association of the real valued decision variable $y_{m,n}(S_i)$ and the binary variable $d_{m,n}(S_i)$ have to be modelled and the decision variables of the optimization problem have to be defined:

$$y_{m,n}(S_i) - d_{m,n}(S_i) \le 0, \forall n \in \mathcal{N}, \forall m \in \mathcal{M}, \forall S_i \in \mathcal{S} \tag{4.8}$$

$$d_{m,n}(S_i) - y_{m,n}(S_i) < 1, \forall n \in \mathcal{N}, \forall m \in \mathcal{M}, \forall S_i \in \mathcal{S} \tag{4.9}$$

$$x_n(S_i) \in \{0, 1\}, \forall n \in \mathcal{N}, \forall S_i \in \mathcal{S} \tag{4.10}$$

$$x_n(S_i) \in \{0, 1\}, \forall n \in \mathcal{N}, \forall S_i \in \mathcal{S} \tag{4.11}$$

$$y_{m,n}(S_i) \ge 0, \forall n \in \mathcal{N}, \forall m \in \mathcal{M}, \forall S_i \in \mathcal{S} \tag{4.12}$$

$$d_{m,n}(S_i) \in \{0,1\}, \forall n \in \mathcal{N}, \forall m \in \mathcal{M}, \forall S_i \in \mathcal{S} \qquad (4.13)$$

The constraints (4.8) and (4.9) incorporate an `if-then-else` constraint, i.e. if a seller $m$ partially allocates a bundle $S_i$ to a buyer $n$ ($y_{m,n}(S_i) > 0$), the binary variable $d_{m,n}(S_i)$ has to be 1 (constraint (4.8)); otherwise, $d_{m,n}(S_i)$ has to be 0 (constraint (4.9)). Finally, the constraints (4.10) - (4.13) define the decision variables of the optimization problem.

As there may exist multiple optimal solutions, ties are broken in favour of maximizing the number of traded bundles and then at random.

**Preliminary Complexity Analysis:** The winner determination model (WDP) is $\mathcal{NP}$-complete. Consider the special case of the WDP with multiple buyers, one seller, no quality attributes and no coupling and maximal divisibility restrictions. In this case, the problem is equivalent to the combinatorial allocation problem (CAP) formulated in [RPH98]. The CAP is equivalent to the set-packing problem (SPP) on hypergraphs [RPH98] which is known to be $\mathcal{NP}$-complete [Kar72]. As CAP can be reduced to the WDP, the WDP is also $\mathcal{NP}$-complete.

Nevertheless, the application of such a complex problem seems to be promising, as the number of different bundles in the resource market is restricted. Moreover, [SSGL02, San99a] report that winner determination problems can be solved quite efficient using modern information technology.

Furthermore, the central market mechanism leads to (approximately) efficient results. As such, it fulfills the project requirements to serve as an economic benchmark for the decentralized market.

**Example (Winner Determination):** Suppose, there are four buyers $n_1, n_2, n_3, n_4$, three sellers $m_1, m_2, m_3$ and two services $g_1 = \{Computation\}$ and $g_2 = \{Storage\}$ with $\mathcal{G} = \{g_1, g_2\}$ each with single attributes, namely $a_{g_1,1} = \{Speed\}$ and $a_{g_2,1} = \{Size\}$. The buyers and sellers can submit bids on the bundles $S_1 = \{g_1\}$, $S_2 = \{g_2\}$, and $S_3 = \{g_1, g_2\}$. Each buyer and seller submits a set of bids which are shown in table 4.3 and 4.4.

| $\mathcal{N}$ | $S_i$ | $v_n(S_i)$ | $q_n(S_i, g_k, a_{g_k,j})$ | $\gamma_n(S_i, g_k)$ | $\varphi_n(S_i, g_i, g_j)$ |
|---|---|---|---|---|---|
| $n_1$ | $S_2$ | 2 | $g_2 \to 15$ | $g_2 \to 1$ | |
| $n_2$ | $S_3$ | 2 | $g_1 \to 500, g_2 \to 15$ | | $g_1, g_2 \to 1$ |
| $n_3$ | $S_1$ | 3 | $g_1 \to 400$ | | |
| $n_4$ | $S_3$ | 2 | $g_1 \to 300, g_2 \to 25$ | $g_1 \to 2$ | |

Table 4.3: Bids of the buyers.

The optimal solution of the winner determination problem is an allocation of the bundles $S_3$ and $S_1$ to the buyers $n_2$, $n_3$, and $n_4$ with $x_{n_2}(S_3) = 1$, $x_{n_3}(S_1) = 1$, and $x_{n_4}(S_3) = 1$. Taking the sellers' allocations (table 4.5) into account, the maximized value

| $\mathcal{M}$ | $S_i$ | $r_m(S_i)$ | $q_m(S_i, g_k, a_{g_k,j})$ |
|---|---|---|---|
| $m_1$ | $S_3$ | 1 | $g_1 \to 500; g_2 \to 40$ |
| $m_2$ | $S_2$ | 3 | $g_1 \to 1000$ |
| $m_3$ | $S_3$ | 2 | $g_1 \to 700; g_2 \to 20$ |

Table 4.4: Bids of the sellers.

$V^*$ of the clearing process can be calculated with $V^* = 4.67$. The corresponding schedule for this allocation is given in table 4.5.

| $\mathcal{M}$ | $S$ | Allocation |
|---|---|---|
| $m_1$ | $S_3$ | $n_4 : g_1 \to 310, g_2 \to 25$ |
| $m_2$ | $S_1$ | $n_3 : g_1 \to 400$ |
| $m_3$ | $S_3$ | $n_2 : g_1 \to 525, g_2 \to 15$ |

Table 4.5: Allocation for the example.

For instance, buyer $n_1$ receives the bundle $S_3 = \{g_1, g_2\}$ from seller $m_4$. Although $n_1$ does not require all the allocated computation capabilities of the allocated resource $g_1$, a partial execution of the specific resource $g_1$ as bundles can only be partial executed as a whole. This stems from the economic fact, that complementary resources in a bundle cannot be priced separately and – as such – an isolated resource cannot be executed partially.

The outcome of this model is allocative efficient, as long as buyers and sellers reveal their valuations truthfully. The incentive to set bids according to the valuation is induced by an efficient pricing mechanism that is introduced in the following.

**Pricing**

The question how to determine payments made by participants to the exchange and vice versa after the mechanism has determined the winners is referred to as pricing problem [PKE01]. With respect to the objective of achieving an efficient allocation, a pricing scheme based on a Vickrey-Clarke-Groves (VCG) [Vic61, Cla71, Gro73] mechanism would attain this objective. Moreover, Groves mechanisms are the only allocative-efficient and incentive compatible mechanisms [GL77].

The basic idea of a VCG mechanism is to grant a participant a discount on its bids. This discount reflects the impact of that bid on the social welfare. A VCG mechanism is efficient, incentive-compatible, and individual rational for participants with quasi linear utility functions [PKE01]. However, [MS83] proved that it is impossible to design an exchange, which is incentive compatible, (interim) individually rational, and budget balanced that achieves efficiency in equilibrium.

Hence, the VCG pricing scheme is briefly illustrated to serve as a benchmark. Sub-

| Agent | $(V_{-w})^*$ | $\Delta_{VICK,w}$ | $p_{VICK,w}(S_i)$ |
|:-----:|:-----:|:-----:|:-----:|
| $n_2$ | 3.15 | 1.5 | $p_{VICK,m_2}(S_3) = 1.5$ |
| $n_3$ | 2.87 | 1.8 | $p_{VICK,m_3}(S_1) = 1.2$ |
| $n_4$ | 3.8 | 0.87 | $p_{VICK,m_4}(S_3) = 1.13$ |
| $m_1$ | 3.3 | 1.37 | $p_{VICK,m_1}(S_3) = 2$ |
| $m_2$ | 2.87 | 1.8 | $p_{VICK,m_2}(S_1) = 3$ |
| $m_3$ | 3.8 | 0.87 | $p_{VICK,m_3}(S_3) = 2.38$ |

Table 4.6: Vickrey discounts and prices

sequently, an adapted $k$-pricing scheme is introduced as an adequate compromise mechanism achieving a fairly efficient allocation that is budget balanced.

**Vickrey Pricing**  Let $\mathcal{N}'$ be the set of buyers and $\mathcal{M}'$ be the set of sellers who are part of the allocation (i.e. $x_n(S_i) = 1$ with $n \in \mathcal{N}'$ and $y_{m,n}(S_i) > 0$ with $m \in \mathcal{M}'$). The union of both sets is defined as $\mathcal{W} = \mathcal{N}' \cup \mathcal{M}'$, where $w \in \mathcal{W}$ is an agent who is part of the allocation.

Let $V^*$ be the maximized value of the clearing mechanism and $(V_{-w})^*$ be the maximized value of the allocation without the agent $w$. Therefore, the Vickrey discount for an agent $w$ can be calculated by $\Delta_{VICK,w} = V^* - (V_{-w})^*$.

In consideration of the Vickrey discounts $\Delta_{VICK,w}$, the Vickrey price $p_{VICK,n}(S_i)$ for a bundle $S_i$ and a buyer $n$ can be calculated by

$$p_{VICK,n}(S_i) = v_n(S_i) - \Delta_{VICK,n},$$

and the Vickrey price $p_{VICK,m}(S_i)$ for a bundle $S_i$ and a seller $m$ by

$$p_{VICK,m}(S_i) = r_m(S_i) \sum_{n \in \mathcal{N}} y_{m,n}(S_i) + \Delta_{VICK,m}.$$

Applying the VCG pricing schema to the above presented example ($V^* = 4.67$) results into the prices $p_{VICK,w}(S_i)$ and the discounts $\Delta_{VICK,w}$ shown in table 4.6.

A Vickrey-Clark-Groves mechanism is efficient and individual rational, however, in an exchange not budget balanced. Aggregating the net payments of the example leads to a negative value with $1.2 + 1.13 + 1.5 - (2 + 3 + 2.38) = -3.55$. In this case, the auctioneer has to endow the exchange, which is practical not realizable (cf. section 4.1.2).

Relaxing the efficiency property, a possible implementation of a budget-balanced pricing rule for double auctions is the $k$-pricing scheme[10].

---

[10]Holding most of the Vickrey-Clarke-Groves properties, another possible pricing schema is the so-called approximated Vickrey pricing mechanism introduced by [PKE01]. Due to complexity issues, we restrict ourselves to the $k$-pricing scheme. An application of the approximated Vickrey pricing mechanism for the Grid can be found in [SNVW05a]

**K-Pricing** The underlying idea of the $k$-pricing scheme is to determine prices for a buyer and a seller on the basis of the difference between their bids [SW93]. For instance, suppose a buyer wants to buy a computation service for 5 and a seller wants to sell a computation service for at least 4. The difference between these bids is $\pi = 1$, i.e. $\pi$ is the surplus of this transaction and can be distributed among the participants.

For a single commodity exchange, the $k$-pricing scheme can be formalized as follows: Let $v_n(S_i) = a$ be the valuation of a buyer $n$ and $r_m(S_i) = b$ be the reservation price of the buyer's counterpart $m$. It is assumed that $a \geq b$, i.e. the buyer has a valuation for the commodity which is at least as high as the seller's reservation price. The price for a buyer $n$ and a seller $m$ can be calculated as $p(S_i) = ka + (1 - k)b$ with $0 \leq k \leq 1$. In consideration of the above mentioned bids for the computation service $g_1$, the price results in $p(g_1) = 0.5 * 5 + (1 - 0.5) * 4 = 4.5$ when using $k = 0.5$.

As in the service market, a balanced order book is assumed, i.e. the number of buyers equals the number of sellers. Therefore, $k$ is set to $k = 0.5$ as it privileges neither the buyers' side nor the sellers' side [Fri91].

The $k$-pricing schema can be adapted for a multi-attribute combinatorial exchange: If a bundle $S_i$ is allocated from one or more sellers, the surplus generated by this allocation is distributed among a buyer and the sellers. Suppose a buyer $n$ receives a computation service $S_1 = \{g_1\}$ with 1000 MIPS in time-slot 4 and values this slot with $v_n(S_1) = 5$. The buyer obtains the computation service $S_1 = \{g_1\}$ by a co-allocation from seller $m_1$ (400 MIPS) with a reservation price of $r_{m_1}(S_1) = 1$ and from seller $m_2$ (600 MIPS) with $r_{m_2}(S_1) = 2$. The distributable surplus of this this allocation is $\pi_n(S_1) = 5 - (1 + 2) = 2$. Buyer $n$ gets $k * \pi_{n,4}(S_1)$ of this surplus, i.e. the price buyer $n$ has to pay for this slot is $p_{k,n}(S_i) = v(S_1) - k\pi_n(S_1)$. Furthermore, the sellers have to divide the other part of this surplus, i.e. $(1 - k)\pi_n(S_1)$. This will be done under consideration of each proportion a seller's bid has on the surplus. In the example, this proportion $\varrho_{m,n}(S_i)$ for seller $m_1$ is $\varrho_{m_1,n}(S_1) = \frac{1}{3}$ and for seller $m_2$ $\varrho_{m_2,n}(S_1) = \frac{2}{3}$. The price for a seller $m$ gets is consequently calculated as

$$p_{k,n}(S_i) = r(S_1) + (1 - k)\pi_n(S_1)\varrho_{m_1,n}(S_1).$$

Extending this scheme to all sellers results in the following formalization: Let $\pi_n(S_i)$ be the surplus for a bundle $S_i$ of a buyer $n$ with all corresponding sellers leads to:

$$\pi_n(S_i) = x_n(S_i)v_n(S_i) - \sum_{m \in \mathcal{M}} y_{m,n}(S_i)r_m(S_i). \tag{4.14}$$

The price for a buyer $n$ is subsequently calculated as

$$p_{k,n}(S_i) = x_n(S_i)v_n(S_i) - k\pi_n(S_i). \tag{4.15}$$

This means that the difference between the valuation ($v_n(S_i)$) of the bundle $S_i$ and the $k$-th proportion of the sum over all corresponding surpluses is determined.

The price of a seller $m$ is calculated in a similar way: First of all, the proportion $\varrho_{m,n}(S_i)$ of a seller $m$ allocating a bundle $S_i$ to the $n$ is given by

$$\varrho_{m,n}(S_i) = \frac{y_{m,n}(S_i)r_m(S_i)}{\sum_{m \in \mathcal{M}} y_{m,n}(S_i)r_m(S_i)}. \tag{4.16}$$

Having computed $\pi_n(S_i)$ and $\varrho_{m,n}(S_i)$, the price a seller receives for a bundle $S_i$ is calculated as:

$$p_{k,m}(S_i) = \sum_{n \in \mathcal{N}} y_{m,n}(S_i)r_m(S_i) + (1-k) \sum_{n \in \mathcal{N}} \varrho_{m,n}(S_i)\pi_n(S_i). \tag{4.17}$$

Applying this pricing scheme to the above presented example results in the prices in table 4.7. In this case, the exchange does not have to subsidize the participants as it fulfills the budget balance property in a way that no payments towards the mechanism are necessary. As such, the application of the $k$-pricing schema to the resource market is deemed promising. In [SNVW05b] it is shown, that the $k-$ pricing schema may lead to approximate efficient solutions.

| $\mathcal{N}$ | $v_w(S_i)$ | $p_{k,w}(S_i)$ | $\mathcal{M}$ | $r_w(S_i)$ | $p_{k,w}(S_i)$ |
|---|---|---|---|---|---|
| $n_2$ | 3 | 2.25 | $m_1$ | 0.62 | 1.31 |
| $n_3$ | 3 | 2.1 | $m_2$ | 1.2 | 2.1 |
| $n_4$ | 2 | 1.31 | $m_2$ | 1.5 | 2.25 |

Table 4.7: Prices using $k$-Pricing with $k = 0.5$.

**Continuous and Periodic Trading** In analogy to the central service market specified in section 3.3.2, a key consideration is the timing of the clearing process. Again, the mechanism can be cleared continuously or periodically. As both clearing concepts can be easily implemented into a software system, the final decision concerning the clearing of the market will be made during the simulation runs[11].

**Implementation**

The presented clearing mechanism as well as the different pricing approximations are implemented as a Java based prototype [SNVW05a]. CPLEX 9.1[12] is used as an optimization engine for solving the linear mixed integer program. As a consequence, exact and optimal solutions are obtained.

Figure 4.5 depicts briefly the sequence of an auction for the CATNETS scenario. Participants submit their bids in form of a WS-Agreement offer to the market. After that,

---

[11]This can be simple achieved by a configuration setting.

[12]CPLEX `http://www.ilog.com/` is commercial product and is currently the state of the art optimization engine. (Accessed: 22.08.2005)
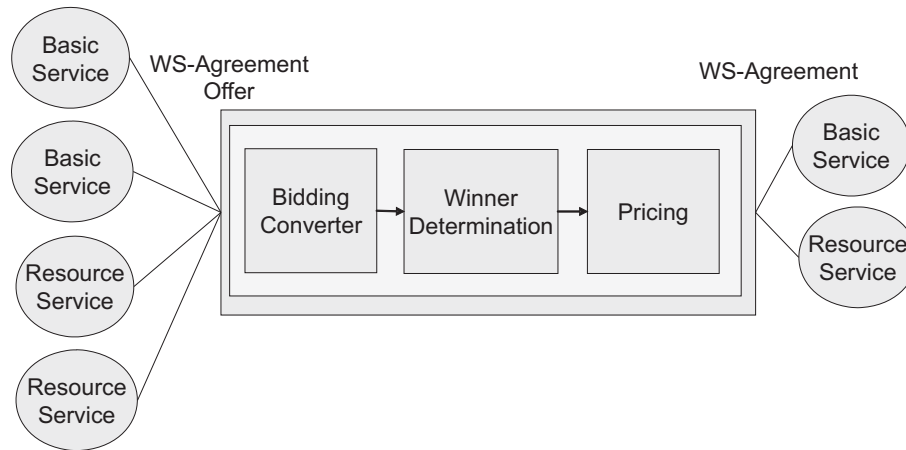
Figure 4.4: Sequence of an auction for the resource market.

the WS-Agreement bids are transformed to an internal representation form. Then, the winner determination component computes the allocation. Finally, prices are computed in consideration of the allocation. As a result of the market mechanism, the participants get informed by means of a WS-Agreement whether they are part of the allocation or not.

Figure 4.5 sketches briefly the implemented main components of the market mechanism. The central component is the `Market` class which instantiates an `Environment` (storing participants, goods, and bundles), an `Orderbook` (storing bids), and a `Mechanism` (implementing the winner determination and pricing mechanism). The `Environment` and the `Orderbook` can be filled by XML based documents or distributions. The `Mechanism` encapsulates the market mechanism by instantiating an `Outcome` class and a `Pricing` class. The `Outcome` class is responsible for the winner determination problem and uses a specific solver (e.g. `CPLEXAdapater` ). The `Pricing` class instantiates a pricing mechanism (e.g. `VCGPricing`) that determines the net payments.

### 4.3.3 Decentralized Market

The heuristic strategy in the decentralized market is similar to the service market. We will describe only the differences to the concept on the service market. The differences arise with the multi-attributive good and the co-allocation on the resource market compared to the service market.

The resource market defines a multi-attributive good, but there is only one single attribute (price) negotiable. Therefore, we can reduce the complexity of the negotiation to a single attributive bilateral negotiation like on the service market.

The handling of bundling and co-allocation is done before the negotiation starts. As on the service market, a buyer has his own local order book, where he collects possible
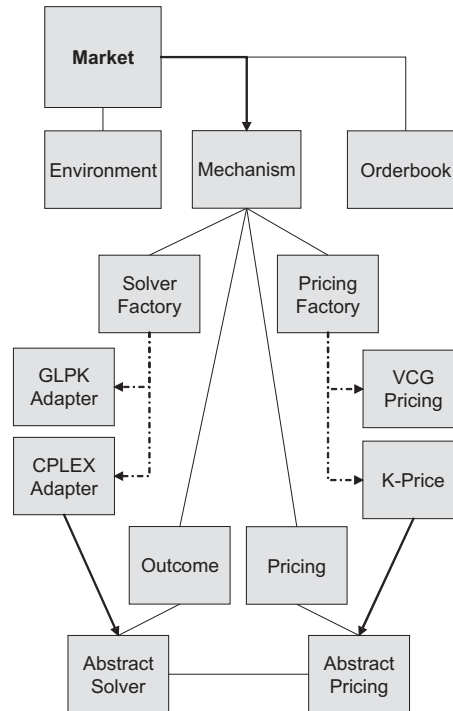
Figure 4.5: Implemented components of the resource allocation mechanism.

bids and ranks them according to his price (see figure 4.6).

However, he cannot rank them using only the price attribute, because the goods are multi-attribute. This means, he has to compute the utility of every item in his order book and rank the incoming offers using a utility function. Before we will describe the utility function, we describe the request strategy of the buyer (basic service).

The buyer needs a specific resource bundle and has a budget for this bundle. He broadcasts his query on the network and collects the offers. A buyer interprets the request received and generates an offer of the whole requested resource bundle or parts of the bundle, if the bundle is allowed to split. The buyer receives lots of offers for his request, either offering the complete resource bundle or parts of this bundle. The problem is, that he has to rank these offers distinguishing in amount and price. This is done using his utility function. An example for this process shows figure 4.7.

In CATNETS, we will use at the beginning a simple utility function which computes the price per basic bundle unit. This basic bundle unit is predefined on the CATNETS resource market. At the end of this step we have a list of possible sellers ranked with the expected utility of the buyer. The next step is to define one bundle out of the offered bundles which fulfills the buyer's request. An optimal solution of this process is known as a very time-consuming operation. Therefore, we will use the heuristic based on fuzzy logic and priority scheduling strategies which are mentioned at the related work section of this chapter. At this stage of the project, we have not finally decided on the scheduling
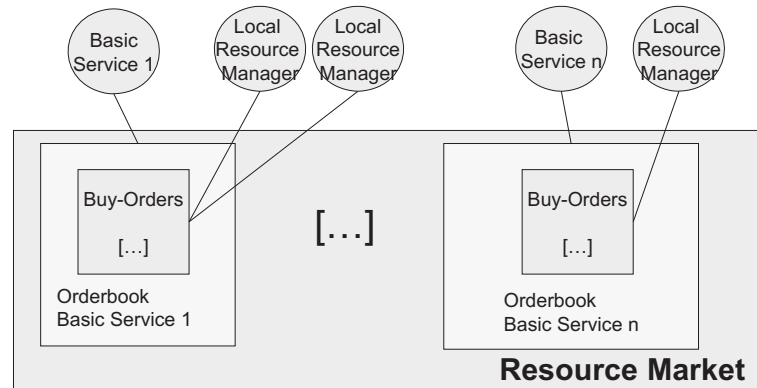
Figure 4.6: Local order books on the resource market.
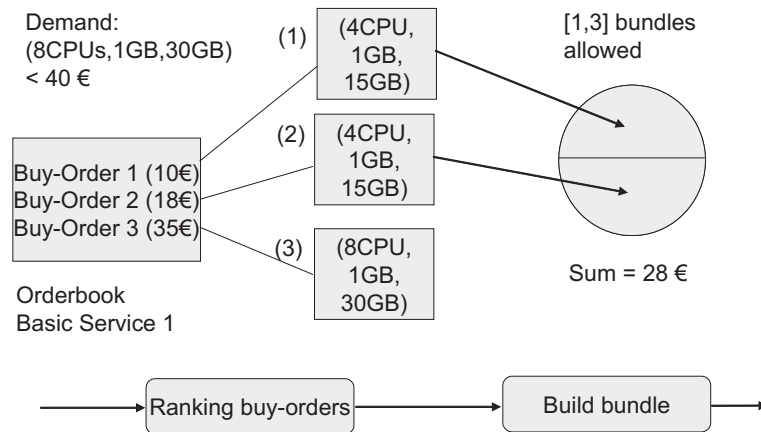
## Bundeling



Figure 4.7: Ranking and bundling on the resource market.

strategies because we decide this issue after analyzing first results on bundling from the simulator.

An example with a first-come-first-serve strategy:

We start with the bundle with the highest expected utility and fill up the bundle with fitting, high-utility resource bundles until we have the requested bundle size. If there is no bundle which fits the remaining resource bundle size, we can choose a bundle which has bigger size to prevent a deadlock of the algorithm. We cut the bundle down to the requested bundle size, because the buyer would only pay for the requested resource bundle size. Afterwards we can start the negotiation of the part-bundles.

**Negotiation protocol**

The basic service breaks up and translates complex requests to resource layer and the resource co-allocator and starts the negotiation with several local resource managers. The local resource manager analyzes the request and creates an offer. This process iterates until an agreement (accept) or reject is reached. If an accept is reached, the resource allocator confirms and informs the basic service about the contracted resources. The basic service continues the negotiation with the complex service, using the information from contracting the resources. The negotiation on the resource layer is processed only once. It is impossible to renegotiate a resource contract. The resuming negotiation with the complex service uses the same negotiation protocol and after an accept the payment process is initiated which pays the basic service and the resource. A reject on the service market will lead to a reject on the resource market.

The negotiation protocol on the resource market is similar as on the service market. The message types are the same, only the bidding language has changed. We will present here the integration of the two markets with the negotiation protocol. Figure 4.8 shows the protocol for both markets, the service and the resource market.

At the beginning of the negotiation, the basic service generates the first proposal to the complex service without complete knowledge of the current market price. However, he estimates the market price using historic information about former negotiations. The basic service does not enter a phase of resource negotiation, because it would be too much overhead for all basic services contracting resources not knowing if the complex service will select this basic service for negotiation.

Once the complex service receives the estimated prices of basic services, he selects one basic service for negotiation. Before the negotiation continues on the service market, the basic service will buy the resources on the resource market. If he succeeds and gets the requested resources, the negotiation on the service market is continued. The protocol does not allow to re-negotiate resources for performance reasons. If the negotiation on the service market fails, also the contract on the resource market will not be confirmed any more. If the negotiation comes to a successful end, the reserved resources are confirmed. A final confirm messages signals the complex service, that he can now be deployed.
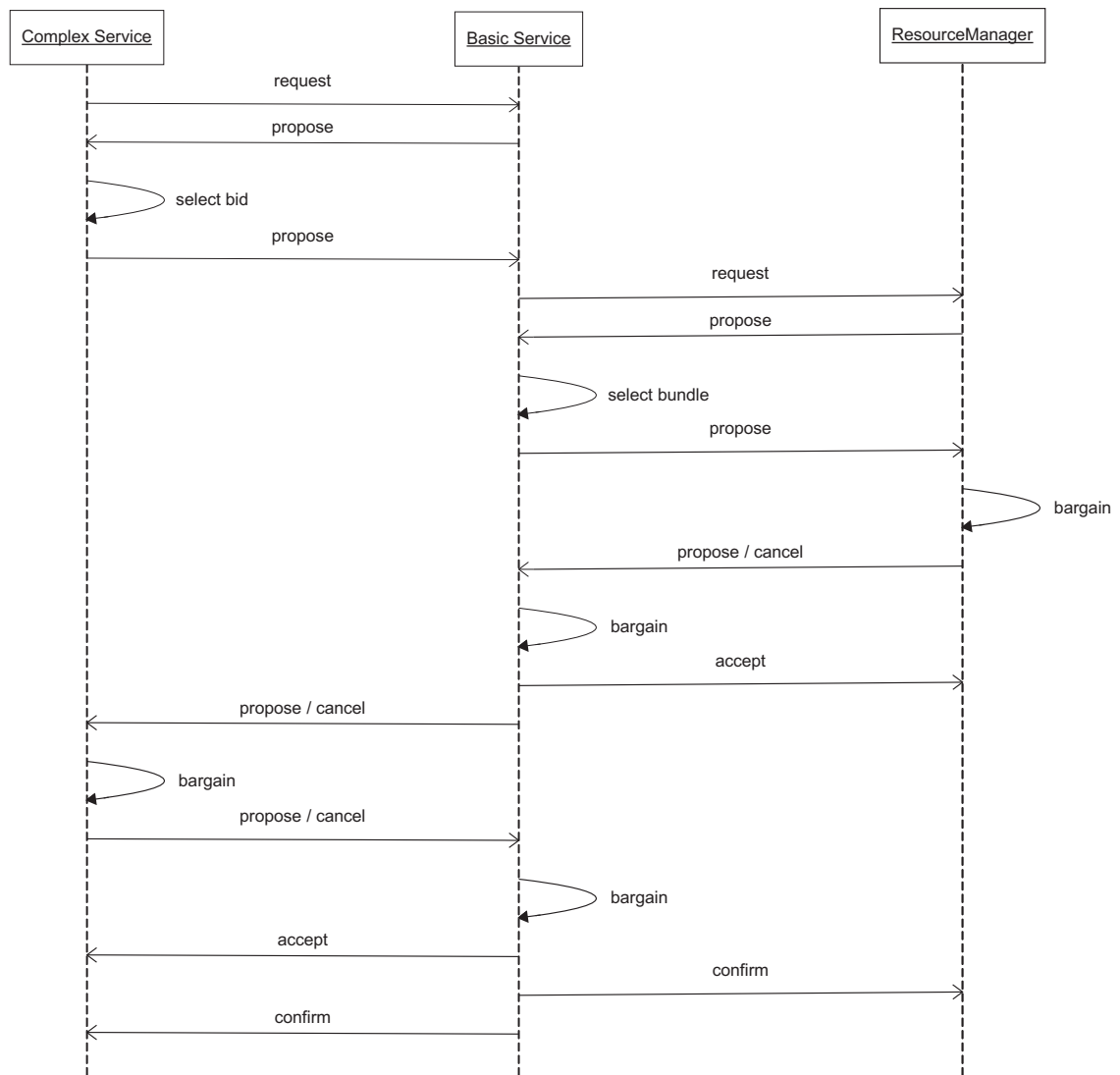
Figure 4.8: The bargaining protocol on jointly both market.

The transformation of the demand on the service market to a request on the resource market regarding quality of service and budget is also an important issue in the connection of the two markets. As we already described in section 3, we designed standardized services on the service market. One service type (e.g. a "gold" PDF converter) represents a discrete quality of service level. This QoS is modeled using templates of the service market. In the CATNETS project we use one template which represents the bidding language on the resource market. Possible implementations can use XML (WS-Agreement) or binary Java objects for representation. We assume here, that the complex services as requestors of basic services know the needed basic service type, that represent their QoS requirements, in advance. A complex service can only request a discrete, pre-defined QoS level. This is introduced to keep the complexity on the service market low.

## 4.4   Relations to other Work Packages

In this chapter, centralized and decentralized resource allocation mechanisms for Grid and ALN markets have been introduced. Both, centralized and decentralized mechanisms influence the implementation of the simulator and the middleware[13] (work package 2 and 3). Their outcomes must also be measurable by the metrics developed in work package 4. The prototype is affected only by the decentralized mechanisms. In the following sections some insights are given how these influences will affect future work in the next months.

### 4.4.1   Relation to Simulator (WP2)

The simulator needs to take inputs from both, centralized and decentralized market mechanisms. Therefore, all bidding and bargaining languages that are developed throughout this chapter have to be included to simulate the resource allocation mechanisms.

Allocation and pricing schemes will be integrated in close cooperation with WP2 in the next period of the project. Especially, the mapping of the environmental definition, the interfaces for the communication between the participants and the transfer of the resource market to the Grid nodes in the simulator are of high relevance.

### 4.4.2   Relation to Proof of Concept (WP3)

Within the proof of concept work package the prototype and the catallactic middleware are developed. Only the decentralized market mechanism is implemented in the catallactic middleware, because the effort of implementing both would be too high.

The key issue is that the bidding languages and the mechanisms are implementable in

---

[13]Note, that the centralized mechanism will only be evaluated by the simulator.

the middleware and the prototype. Beyond this, the interfaces of communication must be maintained. It would be ideal to have homogeneous definitions, communication language and interfaces with the simulator work package in order to combine the implementations.

### 4.4.3 Relation to Evaluation (WP4)

The primary idea of the evaluation work package concerning the mechanisms for the resource market is to measure the economic and technical efficiency of the proposed resource allocation mechanisms (using the measures also described in Section 3.4). Since the goal of the CATNETS project is to compare the efficiency of those, the measures developed in WP4 must be applicable to the developed mechanisms.

From the theoretical perspective this means, that the outcomes must be measurable wrt their economic factors they produce. From the practical point of view the following issues are of high relevance:

- Which data will be produced by the mechanisms during simulation.

- Which data are stored and evaluated after simulations.

- Are the proposed metrics in WP4 capable of investigating these datasets.

- How can the proposed metrics be implemented to test the different efficiency criteria.

Since the results of these measurements will be considered the main outcome of the project, they must derive special attention by all project partners. They have to be carefully developed and adapted to the Grid/ALN market requirements formulated in this chapter.

# Chapter 5

# Mappable Applications

This section compares the CATNETS market model, and the component modules identified so far, to existing applications with a great number of connected peers. All of these applications exhibit the unfavorable properties presented in the previous sections and require efficient coordination mechanisms. By introducing the CATNETS market model, we aim to ameliorating the performance. Considering possible application domains, the following 3 systems have been identified: BITTORRENT, PLANETLAB and CORAL. They will be presented shortly and an analysis is given on the possible matching of Catallaxy to those applications. This matching will be done on the application layer. Lower Network layers are not explicitly mentioned.

The criteria for selection of these applications has been the following: for all potential applications, we have selected applications which are exemplary for their application area, have a certain degree of popularity, and the source code is available for inspection, modification and experimentation. Ideally, the candidate applications should have been evaluated and characterized in public papers, thus we could then compare our results with an external baseline evaluation.

## 5.1 BitTorrent

In the context of Peer-to-Peer networks, we have selected a P2P protocol which has a clearly specified protocol, that is popular enough, and that is used for clearly useful and legal purposes (some other P2P networks are almost only used for sharing copyrighted content). This protocol is BitTorrent [1][Coh03].

With BitTorrent, when multiple people are downloading the same file at the same time, they are also uploading pieces of the file to each other. This redistributes the cost of upload to downloaders, thus making hosting a file with a potentially unlimited number of

---

[1]http://bittorrent.com

downloaders affordable. Following [IUKB$^+$04], a torrent consists of a central component, called tracker and all the currently active peers. BitTorrent distinguishes between two kinds of peers depending on their download status: clients that have already a complete copy of the file and continue to serve other peers are called seeds; clients that are still downloading the file are called leechers. The tracker is the only centralized component of the system. The tracker is not involved in the actual distribution of the file; instead, it keeps meta-information about the peers that are currently active and acts as a rendezvous point for all the clients of the torrent.

A user joins an existing torrent by downloading a torrent file (usually from a Web server), which contains the IP address of the tracker. Generic or specialized web search engines usually lead to pages where a file can be downloaded from one or several trackers. The user has to select one torrent file (and thus the tracker) to start downloading the file which will let him connect to the tracker and an initial seed with a complete copy of the file. In case of multiple trackers available for the same object, statistics about every tracker are published to help the visitor choose the right tracker. To update the tracker's global view of the system, active clients periodically (every 30 minutes) report their state to the tracker or when joining or leaving the torrent. Upon joining the torrent, a new client receives from the tracker a list of active peers to connect to.

Typically, the tracker provides 50 peers chosen at random among active peers while the client seeks to maintain connections to 20-40 peers. If ever a client fails to maintain at least 20 connections, it reconnects the tracker to obtain additional peers. The set of peers to which a client is connected is called its peer set. The clients involved in a torrent cooperate to replicate the file among each other using swarming techniques: the file is broken into equal size chunks (typically 256kB each) and the clients in a peer set exchange chunks with one another. The swarming technique allows the implementation of parallel download where different chunks are simultaneously downloaded from different clients. Each time a client obtains a new chunk, it informs all the peers it is connected with.

Interactions between clients are primarily guided by two principles. First, a peer preferentially sends data to peers that reciprocally sent data to him. This "tit-for-tat" strategy is used to encourage cooperation and ban "free-riding". Second, a peer limits the number of peers being served simultaneously to 4 peers and continuously looks for the 4 best downloaders (in terms of the rate achieved) if it is a seed or the 4 best uploaders if it is a leecher. In terms of the CATNETS model, people interested in downloading a file, running a web browser and a BitTorrent client has the role of Client. They look for a torrent file (a tracker) on a search engine and looking at the statistics of several trackers offering the same file they manually select one tracker (on the Service market and the tracker has the role of Basic Service). The tracker joins the client in a swarm of peers exchanging fragments of the file of common interest. All BitTorrent clients in the swarm belong to the Resource market and are acting as Resources.

In terms of the CATNETS model, people interested in downloading a file, running a web browser and a BitTorrent client has the role of a Complex Service. They look for a

torrent file (a tracker) on several online web catalogues and look at the statistics of several trackers offering the same file. They manually select one tracker (on the service market). The tracker adds the requesting complex service to a swarm of peers (resource services) exchanging fragments of the file of common interest. All BitTorrent clients in the swarm belong to the resource market and are acting as resource services.

## 5.2 PlanetLab

In the context of ALN for distributed computing using computing resources across administrative boundaries (Grid computing), we have identified two unique initiatives offering an open infrastructure for the deployment of services and the use of computational resources in the academic or industrial environment. Both are unique in terms of size, availability and relative maturity. They are PlanetLab [CCR+03] [BBC+04a] and Globus [2].

PlanetLab is a geographically distributed overlay network designed to support the deployment and evaluation of planetary-scale network services. Two high-level goals shape its design. First, to enable a large research community to share the infrastructure, PlanetLab provides distributed virtualization, whereby each service runs in an isolated slice of PlanetLab's global resources. Second, to support competition among multiple network services, PlanetLab decouples the operating system running on each node from the network-wide services that define PlanetLab, a principle referred to as unbundled management.



Figure 5.1: PlanetLab map of member organizations (as of 9/2005)

PlanetLab currently includes over 500 machines spanning 250 sites (i.e. organizations) and more than 20 countries 5.1. It supports more than 500 research projects which

---

[2]http://www.globus.org

focus on a wide range of services, including file sharing and network embedded storage, content distribution networks, routing and multi-cast overlays, QoS overlays, scalable object location services, anomaly detection mechanisms, and network measurement tools. The PlanetLab middleware and API is open and extensible. Distributed applications can use the services offered by the virtualized operating system in each node (currently the Linux API) plus the XML-RPC interface offered by the PlanetLab Central (PLC) administration.

The service-resource cycle in PlanetLab is as follows:

- In every node, the node manager is in charge of creating and allocating resources to vservers (virtual machines), and the resource monitor is in charge of tracking node's availability of resources and informing the central agent about available resources."

- The agent tracks nodes' free resources, which are advertised to resource brokers and offered as tickets to services interested in acquiring and using resources. This agent is part of Planet-Lab Central (PLC), a centrally-controlled brokerage service that can be decentralized using a delegation mechanism.

- In every service, the resource broker obtains tickets from agents on behalf of service managers, which are in charge of redeeming tickets with node managers to acquire resources, and if resources can be acquired, start the service in that node.

In terms of the CATNETS model, processes interested in using a given service have the role of complex services. They look for and select basic services. All node managers and resource monitors act as resources and all resource brokers and service managers act as basic services on the resource market. Both are mediated by the central PlanetLab agent.

The Globus toolkit is the reference implementation of the standard Grid protocols and APIs that the Global Grid Forum (GGF) is defining for different aspects of distributed computing, such as security, resource management, data management, and information discovery. The Globus middleware has been adopted by most of the Grid projects worldwide. In comparison with the Globus Grid implementation which offers a higher level homogeneous API, PlanetLab offers a less coupled and simpler API based on the idea of virtualization. While the grid offers an ample collection of middleware services unified in a single architecture as exemplified by the Globus, PlanetLab offers an API for the basic service for creating slices, and associating people and nodes to them. Slices appear to users as a set of virtual Linux machines (i.e. offering a multiple Linux API instead of a higher level and abstract API: virtualization in contrast to abstraction). Additionally, there may be competing services providing additional functionality that also run using the PlanetLab infrastructure (multiplicity in contrast to homogeneity). There is another difference to emphasize: While the grid is primarily interesting in gluing together a modest number of high-performance computing resources connected by high performance networks, PlanetLab is focused on scaling less bandwidth and CPU intensive applications

offering innovative services across a wider collection of nodes [PACR02]. Finally, both worlds can be combined: There are pilot experiments where Globus based applications are run on top of PlanetLab (in a slice, on several nodes or slivers).

## 5.3 Coral

In the context of content distribution, the selection criteria applies to two academic content distribution networks (CDNs) which by coincidence both run on the PlanetLab infrastructure: Coral [FFM04] and CoDeeN [PWP+03]. CoDeeN is a proxy based CDN with some restrictions and limitations, and in contrast Coral provides the typical service that a CDN does with some very interesting properties, and focusing on redirecting clients requests to the "best" copy in terms of load, locality, proximity, offloading work from web origin servers. Coral CDN is a decentralized, self-organizing, Peer-to-Peer web-content distribution network 5.2. Coral CDN leverages the aggregate bandwidth of volunteers (typically PlanetLab slivers) running the software to absorb and dissipate most of the traffic of web sites using the system. In doing so, Coral CDN replicates content in proportion to the content's popularity, regardless of the publisher's resources, in effect democratizing content publication [FFM04].
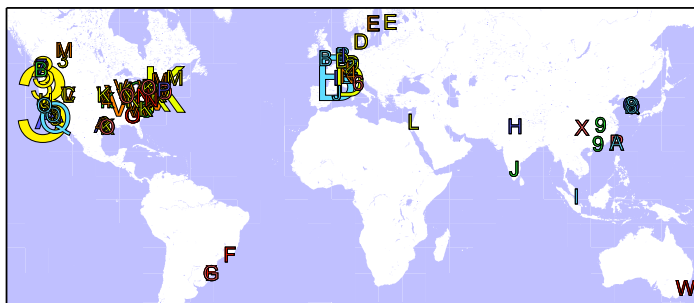


Figure 5.2: Coral's deployment and clusters based on network round-trip-time (letter identifies cluster)

To use Coral CDN, a content publisher - or someone posting a link to a high-traffic portal - simply appends ".nyud.net:8090" to the host name in a URL. Through DNS redirection, oblivious clients with unmodified web browsers are transparently redirected to nearby Coral web caches. These caches cooperate to transfer data from nearby peers whenever possible, minimizing both the load on the origin web server and the end-to-end latency experienced by browsers. This requires two mechanisms: finding a close

peer and identifying a close copy of the requested object. The first is achieved by mapping Coral servers and clients into clusters based on latency. The second is done using a locality-aware request routing algorithm or indexing abstraction (also know as a Distributed Sloppy Hash Table or DSHT). Every Coral peer is running three elements: a DNS server, a HTTP proxy and a DSHT element.

The Coral CDN is implemented on top of a very simple middleware based on RPC over UDP, structured in terms of events and callbacks, with a module for clustering nodes, mapping client locations, routing requests by proximity (Coral DSHT), and modified DNS and HTTP proxy servers.

In terms of the CATNETS model, applications interested in a file use a Coral plug-in to "coralize" URLs. These applications have the role of a complex service. They request a coralized URL, thus going to a Coral DNS server where a response, the IP address of a close-by Coral proxy will be selected among many of them, based on the location of the complex service. This is the service market, the Coral http proxy has the role of a basic service. The complex service Coral plug-in will contact the http proxy with the given IP address. Then the proxy will look for the requested file in its own storage or will look for a close copy of the file in other peers using the Coral DSHT routing algorithm. Proxies belong to the Resource market, the election in the market is determined by the DSHT algorithm looking for a close copy of a file, and proxies are acting as resources.

# Chapter 6

# Conclusion

The key issue of this work package is the structured and theoretical identification and definition of market mechanisms for Application Layer Networks. The results of this deliverable provide the theoretical basis for the simulator work package (WP2), the proof-of-concept work package (WP3), and the evaluation work package (WP4).

Hence, we provide a summary on the results of this report in Section 6.1 and an outlook on the next steps to be taken in Section 6.2.

## 6.1   Review of this Work

This work comprises several parts. Firstly, a market based perspective for Application Layer Networks is drawn. Here, two markets are distinguished: a service and a resource market. The service market is used by customer agents that aim on purchasing a certain service in order to fulfil a task. The second market – the resource market – is the market on which the service instances trade the resources they need in order to operate the offered services (compare Chapter 1).

The approach which is chosen in both cases in order to design the market characteristics is the so-called Market Engineering approach (compare Chapter 2). This approach is based on a process including several steps to define markets in a structured way. The process is accompanied by a set of well chosen methods that are consequently applied in the definition phase of the market mechanisms for ALNs in the CATNETS project. Next to defining and implementing as well as testing market mechanisms, this approach also employs means of experimental analysis and computerized simulation for the evaluation of market mechanisms. An important source is the work done in the domain of mechanism design and economic engineering in past years.

In the following, the Market Engineering approach is employed for the service and the resource market (compare Chapter 3 and Chapter 4). As a result of the analysis of

the requirements for the service market, centralized and decentralized mechanisms are sketched.

## 6.2   Outlook on Next Steps

The next steps within the CATNETS project comprise the implementation of the central and decentral mechanisms into the simulator platform (WP2). Furthermore, the bilateral bargaining strategies (decentral) will be used in the proof-of-concept prototype (WP3) in order to evaluate the application of the Catallaxy concept in Application Layer Networks.

Finally, the central and decentral mechanisms will be evaluated by means of the work done in work package 4.

# Bibliography

[ACD+01]    Nachiket Acharya, Chaitanya Chokkaredd, Pinkeshkumar Desai, Rajesh Devrajan, Michael P. Frank, Sriram Kumar Nallan Chakravarthul, Murali Nagendranath, Pradeep Padala, Heeyong Park, Gargi Sur, Mark Tobias, Chi-Kin Wong, and BongJu Yu. The Open Computation Exchange & Auctioning Network (OCEAN). Technical report, Department of Computer & Information Science & Engineering, University of Florida, 2001.

[ACD+05]    Alain Andrieux, Karl Czajkowski, Asit Dan, Kate Keahe, Heiko Ludwig, Toshiyuki Nakata, Jim Pruyne, John Rofrano, Steve Tuecke, and Ming Xu. Web Services Agreement Specification (WS-Agreement). GWD-R, Global Grid Forum, 2005.

[ACSV04]    Alvin AuYoung, Brent N. Chun, Alex C. Snoeren, and Amin Vahdat. Resource Allocation in Federated Distributed Computing Infrastructures. In *Proceedings of the 1st Workshop on Operating System and Architectural Support for the On-demand IT InfraStructure*, 2004.

[AF99]    O. Arndt and B. Freisleben. Batch Queueing in the WINNER Resource Management System. In *Parallel and Distributed Processing Techniques and Applications, Las Vegas, Nevada*, 1999.

[AG05]    Deutsche Boerse AG. Europe's Premier Listing Platform. Fact Sheet, Deutsche Boerse AG, 2005.

[AH00]    Eytan Adar and Bernardo A. Huberman. Free Riding on Gnutella. *First Monday*, 5(10), 2000.

[AMA04]    F. Azzedin, M. Maheswaran, and N. Arnason. A Synchronous Co-allocation Mechanism for Grid Computing Systems. *Cluster Computing*, 7(1):39–49, 2004.

[ASS02]    K.J. Arrow, A.K. Sen, and K. Suzumura. *Handbook of Social Choice and Welfare*, volume Volume 1. North-Holland, 2002.

[BA00]     R. Buyya and J. Abramson, D.and Giddy. Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. In *Proceedings of the 4th Internationall Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2000),* , 2000.

[BAG01]    Rajkumar Buyya, David Abramson, and Jonathan Giddy. A Case for Economy Grid Architecture for Service Oriented Grid Computing. In *IPDPS '01: Proceedings of the 10th Heterogeneous Computing Workshop at HCW 2001 (Workshop 1)*, page 20083.1, Washington, DC, USA, 2001. IEEE Computer Society.

[BAGS02]   Rajkumar Buyya, David Abramson, Jonathan Giddy, and Heinz Stockinger. Economic Models for Resource Management and Scheduling in Grid Computing. *The Journal of Concurrency and Computation: Practice and Experience*, 14(13-15):1507–1542, 2002.

[BAV04]    Rajkumar Buyya, David Abramson, and Srikumar Venugopal. The Grid Economy. In *Proceedings of the IEEE, Grid Computing*, 2004.

[BBC$^+$04a]   A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak. Operating System Support for Planetary-Scale Network Services. In *First Symposium on Networked Systems Design and Implementation (NSDI)*, pages 253–266, 2004.

[BBC$^+$04b]   Andy Bavier, Mic Bowman, Brent Chun, David Culler, Scott Karlin, Steve Muir, Larry Peterson, Timothy Roscoe, Tammo Spalink, and Mike Wawrzoniak. Operating Systems Support for Planetary-Scale Network Services. In *Proceedings of the First Symposium on Networked Systems and Design and Implementation, San Fransisco*, 2004.

[BCZ92]    T.A. Byrd, K.L. Cossick, and R.W. Zmud. A Synthesis of Research on Requirements Analysis and Knowledge Acquisition Techniques. *MIS Quarterly*, 16(1):117–138, 1992.

[BDGS05]   Ravi Bapna, Sanjukta Das, Rober Garfinkel, and Jan Stallaert. A Market Design for Grid Computing. Technical report, Department of Operations and Information Management, University of Connecticut, 2005.

[BKS00]    S. Bussmann and K. K. Schild. Self-Organizing Manufacturing Control: An In-dustrial Application of Agent Technology. In *Proc. of the 4th International Confer-ence on Multiagent Systems (ICMAS), Boston*, 2000.

[BM02]      Rajkumar Buyya and Manzur Murshed. GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. *The Journal of Concurrency and Computation: Practice and Experience (CCPE)*, 14(13-15), 2002.

[BN04]      Shantanu Biswas and Y. Narahari. An Iterative Auction Mechanism for Combinatorial Logistics Exchanges. In *Proceedings of the 9th International Symposium on Logistics*, 2004.

[Bre96]     T. Brenner. Learning in a Repeated Decision Process: A Variation-Imitation-Decision Model. Papers on Economics & Evolution. Technical report, Jena, Max-Planck-Institut fuer die Erforschung von Wirtschaftssystemen, 1996.

[Bre02]     T. Brenner. A Behavioural Learning Approach to the Dynamics of Prices. *Computational Economics*, 19:67–94, 2002.

[BSGA01]    Rajkumar Buyya, Heinz Stockinger, Jonathan Ghiddy, and David Abramson. Economic Models for Management of Resources in Peer-to-Peer and Grid Computing. In *Proceedings of the International Conference on Commercial Applications for High Performance, 2001*, 2001.

[CAB$^+$94]  Derek Coleman, Patrick Arnold, Stephanie Bodoff, Chris Dollin, Helena Gilchrist, Fiona Hayes, and Paul Jeremaes. *Object-Oriented Development - The FUSION Method*. Prentice Hall, Englewood Cliffs, NJ, 1994.

[CB98]      D. Cliff and J. Bruten. Less than Human: Simple adaptive trading agents for CDA markets. In *Proceedings of the IFAC Symposium on Computation in Eco-nomics, Finance, and Engineering: Economic Systems (CE-FES'98)*, 1998.

[CB02]      Madhu Chetty and Rajkumar Buyya. Weaving Computational Grids: How Analogous Are They with Electrical Grids? . *Computing - Science and Engineering*, 4(4):61–71, 2002.

[CCR$^+$03]  Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. PlanetLab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12, 2003.

[CFK04]     Karl Czajkowski, Ian Foster, and Carl Kesselman. *Resource and Service Management*, volume 2, chapter 18, pages 259–283. Elsevier , 2004. In: The Grid 2 - Blueprint for a New Computing Infrastructure.

[Cla71]     E.H. Clarke. Multipart pricing of Public Goods. *Public Choice*, 2:19–33, 1971.

[Cle96]     Scott H. Clearwater. *Market-based control*. World Scientific, 1996.

[CO00]      H. Casanova and G. Obertelli. The AppLeS parameter sweep template: user-level middleware for the grid. 2000.

[COBW00]    Henri Casanova, Graziano Obertelli, Francine Berman, and Rich Wolski. The AppLeS parameter sweep template: user-level middleware for the grid. In *Proceedings of the 2000 ACM/IEEE conference on Supercomputing*. IEEE Computer Society, 2000.

[Coh03]     Bram Cohen. Incentives Build Robustness in BitTorrent, 2003.

[Con02]     Wolfram Conen. Economically Coordinated Job Shop Scheduling and Decision Point Bidding - An Example for Economic Coordination in Manufacturing and Logistics. In *Proceedings of the 16th Workshop on Planen, Scheduling und Konfigurieren, Entwerfen*, 2002.

[Cra03]     Peter Cramton. Electricity Market Design: The Good, the Bad, and the Ugly. *Proceedings of the Hawaii International Conference on System Sciences*, 2003.

[CS01]      K. Czajkowski and V. Sander. Grid Resource Management Protocol: Requirements. GGF Scheduling Work Group Proposal, University of Southern California, John von Neumann Institut for Computing, 2001.

[CY91]      P. Coad and E. Yourdon. *Object-Oriented Analysis*. Prentice-Hall, Englewood Cliffs, N.J, 1991.

[DHJM99]    Alan R Dennis, Glenda S Hayes, Daniels Jr., and Robert M. Business Process Modeling with Group Support Systems. *Journal of Management Information Systems*, 15(4):115–142, 1999.

[DWL03]     P. Dunne, M. Wooldridge, and M. Laurence. The complexity of contract negotiation. 2003.

[EHY02]     Carsten Ernemann, Volker Hamscher, and Ramin Yahyapour. Economic Scheduling in Grid Computing. In *Proceedings of the 8th International Workshop on Job Scheduling Strategies for Parallel Processing*, pages 128–152. Springer-Verlag, 2002.

[EPS00]     Torsten Eymann, Boris Padovan, and Detlef Schoder. The Catallaxy as a new Paradigm for the Design of Information Systems. In *Proceedings of the 16th IFIP World Computer Congress, Conference on Intelligent Information Processing*, 2000.

[ER98]      I. Erev and A. E. Roth. Predicting How People Play Games - Reinforcement Learning in Experimental Games with Unique, Mixed Strategy Equilibria. *American Economic Review*, 88(4):848–881, 1998.

[ERA⁺03]   T. Eymann, M. Reinicke, O. Ardaiz, P. Artigas, L. Daz de Cerio, F. Freitag, R. Messegue, and D. Navarro, L. Royo. Decentralized vs. Centralized Economic Coordination of Resource Allocation in Grids. In *Proceedings of the 1st European Across Grids Conference*, 2003.

[Eym00]    Torsten Eymann. *AVALANCHE - An Agent-Based Decentralized Coordination Mechanism For Electronic Marketplaces*. PhD thesis, Wirtschaftswissenschaftliche Fakultaet Albert-Ludwigs-Universitt Freiburg, 2000.

[Eym03]    T. Eymann. *Digitale Geschftsagenten*. Springer Xpert.press, Heidelberg, 2003.

[FA03]     Geraldine Fennell and G. M. Allenby. Market Definition, Segmentation, and Positioning: An Integrated Approach. Working Paper, 2003.

[Fau94]    L. V. Fausett. *Fundamentals of Neural Networks*. Prentice Hall, 1994.

[FFM04]    Michael J. Freedman, Eric Freudenthal, and David Mazires. Democratizing Content Publication with Coral. In *1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)*, March 2004.

[FFRS02]   Thomas Friese, Bernd Freisleben, Steffen Rusitschka, and Alan Southall. A Framework for Resource Management in Peer-to-Peer Networks. In Mehmet Aksit and Mira Mezini and Rainer Unland, editor, *NetObjectDays*, volume 2591 of *Lecture Notes in Computer Science*, pages 4–21. Springer, 2002.

[FGG⁺04]   I. Foster, J. Gieraltowski, S. Gose, N. Maltsev, E. May, A. Rodriguez, D. Sulakhe, and the other Grid2003 Project Members. The Grid2003 Production Grid:Principles and Practice. In *Proceedings of the Thirteenth IEEE International Symposium on High-Performance Distributed Computing (HPDC13)*, 2004.

[FK04]     Ian Foster and Carl Kesselman. *The Grid 2 - Blueprint for a New Computing Infrastructure*, volume 2. Elsevier, 2004.

[FNT02]    C. Foster, I.and Kesselman, J.M. Nick, and S. Tuecke. Grid Services for Distributed System Integration. *IEEE Computer*, 35(6):37–46, 2002.

[FR98]     E. G. Furubotn and R. Richter. *Institutions and Economic Theory: The Contribution of the New Institutional Economics*. University of Michigan Press, Detroit, 1998.

[Fri91]      D. Friedman. The Double Auction Market Institution: A Survey. In D. Friedman and J. Rust, editor, *The Double Auction Market - Institutions, Theories, and Evidence*, pages 3–26. Cambridge MA, Perseus Publishing, 1991.

[FS01]       Gianni De Fraja and Jozsef Sakovics. Walras Retrouve: Decentralized Trading Mechanisms and the Competitive Price. *Journal of Political Economy*, 109(4):842–863, 2001. available at http://ideas.repec.org/a/ucp/jpolec/v109y2001i4p842-863.html.

[FSSW02]     Ming Fan, Sayee Srinivasan, Jan Stallaert, and Andrew Whinston. *Electronic Commerce and the Revolution in Financial Markets*. South Western - Thomson Learning, 2002.

[FTF$^+$01]      J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke. Condor-G: A Computation Management Agent for Multiinstitutional Grids. In *International Symposium on High Performance Distributed Computing*, pages 55–67, San Francisco, CA, 2001.

[GAR96]      J. Gehring and A. A. Reinefeld. ARS - A Framework for Minimizing the Job Execution Time in a Metacomputing Environment. *Future Generation Computer Systems*, 12(1):87–99, 1996.

[GL77]       Jerry Green and Jean-Jacques Laffont. Characterization of Satisfactory Mechanisms for the Revelation of Preferences for Public Goods. *Econometrica*, 45(2):427–438, 1977.

[Gol93]      D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Reading MA, 1993.

[Gro73]      T. Groves. Incentives in Teams. *Econometrica*, 41:617–631, 1973.

[GSVW05]     M. Grunenberg, B. Schnizler, D. Veit, and C. Weinhardt. Innovative Handelssysteme fuer Finanzmrkte und das Computational Grid. In *Proceedings der 67. wissenschaftliche Jahrestagung des Verbandes der Hochschullehrer fuer Betriebswirtschaft, Kiel*, 2005.

[HB95]       Karen Holtzblatt and Hugh R. Beyer. Requirements gathering: the human factor. *Communications of the ACM*, 38(5):31 – 32, 1995.

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

[Hol92]      J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence*. MIT Press, Cambridge, 1992.

[Hol04]     Carsten Holtmann. *Organisation von Maerkten - Market Engineering fuer den Wertpapierhandel*. PhD thesis, Economics and Business Engineering. Karlsruhe, Germany, University of Karlsruhe (TH), 2004.

[Hop99]     E. Hoppmann. Unwissenheit, Wirtschaftsordnung und Staatsgewalt. In Vanberg, V., editor, *Freiheit, Wettbewerb und Wirtschaftsordnung*. Haufe Verlag, 1999.

[HR86]      J. D. Hlavacek and N. M. Reddy. Identifying and qualifying industrial market segments. *European Journal of Marketing*, 20(2):8–21, 1986.

[HRRM00]    Cengiz Haksever, Barry Render, Roberta S. Russell, and Robert G. Murdick. *Service Management and Operations*. Prentice-Hall, Upper Saddle River, NJ, 2000.

[Hur72]     Leonid Hurwicz. On informationally decentralized systems. In McGuire, C.B. and Radner, R., editor, *Decision and Organization*. North-Holland Publishing Company, Amsterdam, NL, 1972.

[Hur73]     L. Hurwicz. The Design of Mechanisms for Resource Allocation. *American Economic Review*, 69(2):395–402, 1973.

[IF01]      Adriana Iamnitchi and Ian T. Foster. On Fully Decentralized Resource Discovery in Grid Environments. In *Proceedings of the Second International Workshop on Grid Computing*, pages 51–62. Springer-Verlag, 2001.

[IUKB+04]   M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Hamra, and L. Garces-Erice. Dissecting BitTorrent: Five months in a torrent's lifetime, 2004.

[Jac02]     Matthew O. Jackson. Mechanism Theory. Oxford ,UK, 2002. Eolss Publishers.

[Kar72]     Richard M. Karp. Reducibility among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editor, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[KBM02]     Klaus Krauter, Rajkumar Buyya, and Muthucumaru Maheswaran. A Taxonomy and Survey of Grid Resource Management Systems. *International Journal of Software: Practice and Experience*, 32(2):135–164, 2002.

[KC02]      Chris Kenyon and Giorgos Cheliotis. Architecture Requirements for Commercializing Grid Resources. In *Proceedings of the 11 th IEEE International Symposium on High Performance Distributed Computing HPDC-11 20002 (HPDC'02)*, page 215. IEEE Computer Society, 2002.

[KC03]     J. O. Kephart and D. Chess. A Vision of Autonomic Computing. *IEEE Computer*, 36(1):41–50, 2003.

[KDJN03]   H. Lim Choi Keung, Justin R.D. Dyson, Stephen A. Jarvis, and Graham. R Nudd. Performance Evaluation of a Grid Resource Monitoring and Discovery Service. *IEEE Proceedings - Software*, 150(4):243–251, 2003.

[Keu99]    Frank Keuper. *Fuzzy-PPS-Systems*. PhD thesis, Wirtschaftswissenschaftliche Fakultt Universitt Hamburg, 1999.

[KM05]     Matthias Kunzelmann and Juho Maekioe. Pegged and Bracket Order as a Success Factor in Stock Exchange Competition. In R. A. Fethi and D. J. Veit and C. Weinhardt, editor, *Proceedings of 2nd Conference FinanceCom*, pages 49–56, 2005.

[KP03]     Jayant Kalagnanam and David Parkes. *Supply Chain Analysis in the eBusiness Era*, chapter Auctions, Bidding and Exchange Design. Kluwer Academic Publishing, 2003.

[Kri02]    Vijay Krishna. *Auction theory*. Academic Press, San Diego, 2002.

[KSBE00]   P. Kearney, R.E. Smith, C. Bonacina, and T. Eymann. Integration of Computational Models Inspired by Economics and Genetics. *BT Technology Journal*, 18(4):150–161, 2000.

[KSL04]    G.E. Kersten, S. Strecker, and K.P. Law. Protocols for Electronic Negotiation Systems: Theoretical Foundations and Design Issues. In *Proceedings of the 15th International Workshop on Database and Expert Systems Applications (DEXA)*, 2004.

[KSTT04]   Ramayya Krishnan, Michael D. Smith, Zhulei Tang, and Rahul Telang. The Impact of Free-Riding on Peer-to-Peer Networks. In *Proceedings of the 37th Hawaii International Conference on System Sciences*, 2004.

[LHF04]    Kevin Lai, Bernardo A Huberman, and Leslie Fine. Tycoon: A Distributed Market-based Resource Allocation System, 2004.

[Loe04]    C. Loeser. Demand Forecasting in P2P and GRID Systems. In *Proceedings of the 10th International Conference on Cybernetics and Information Technologies, Systems and Applications (CITSA-2004)*, Orlando, Florida, 2004.

[LW00]     A. R. Lomuscio and M. J. Wooldridge. *A classification scheme for negotia-tion in electronic commerce*. LNAI Springer, Heidelberg, 2000.

[MCM75]    L. Daniel Maxim, Daniel E. Cullen, and John G. Mardo. Optimal Acceptance Test Plans with Grouping. *Technometrics*, 17(3):315–320, 1975.

[Mil04]    Paul Milgrom. *Putting Auction Theory to Work*. Cambridge University Press, 2004.

[MM87]    R.P. McAfee and J. McMillan. Auctions and Bidding. *Journal of Economic Literatur*, 25(2):699–738, 1987.

[MS83]    Robert B. Myerson and Mark A. Satterthwaite. Efficient Mechanisms for bilateral Trading. *Journal of Economic Theory*, 28:265–281, 1983.

[MTE03]    J. Mller and T. T. Eymann. Optimizing Strategy in Agent-based Automated Ne-gotiation. In *Wirtschaftsinformatik 2003*. Dresden, Physica Verlag, 2003.

[Nab96]    Stefan Nabben. *Ciruit Breaker Funktionen und Auswirkungen bedingter Boersenregeln*. Gabler, Wiesbaden, 1996.

[NBC$^+$05]    Chaki Ng, Philip Buonadonna, Brent N. Chun, Alex C. Snoeren, and Amin Vahdat. Addressing Strategic Behavior in a Deployed Microeconomic Resource Allocator. In *In Proceedings of the 3rd Workshop on Economics of Peer-to-Peer Systems*, 2005.

[NC90]    D. C. North and R. Calvert. *Institutions, Institutional Change and Economic Performance - Political Economy of Institutions and Decisions*. Cambridge, Cambridge University Press, 1990.

[Nea96]    R. M. Neal. Bayesian Learning for Neural Networks. *Lecture Notes in Statistics*, 118, 1996.

[Neu04a]    D. Neumann. *Market Engineering - A Structured Design Process for Electronic Markets*. PhD thesis, Economics and Business Engineering. Karlsruhe, Germany, University of Karlsruhe (TH), 2004.

[Neu04b]    Dirk Neumann. Knowledge-Based Auction Design. Working Paper, 2004.

[Neu05]    Dirk Neumann. Market based resource allocation systems – A survey. Working Paper, Chair for Information Management and Systems, University of Karlsruhe (TH), 2005.

[NLRC88]    N. Nisan, S. London, O. Regev, and N. Camiel. Globally Distributed Computation over the Internet - The POPCORN Project . In *Proceedings of the 18th International Conference on Distributed Computing Systems, 26 - 29 May, 1998, Amsterdam, The Netherlands. IEEE Computer Society*, 1988.

[OF99]        James Odell and Martin Fowler. *Advanced object-oriented analysis and design using UML*. Cambridge Univ. Press, Cambridge, 1999.

[Ore01]       Shmuel S. Oren. Market Based Risk Mitigation: Risk Management vs. Risk Avoidance. In *White House OSTP/NSF Workshop on Critical Infrastructure Interdependencies*, Washington DC, 2001.

[PACR02]      L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. *First Workshop on Hot Topics in Networking (HotNets-I)*, 2002.

[Par01]       David Christopher Parkes. *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 2001.

[PB84]        G. Pahl and W. Beitz. *Engineering Design*. The Pitman Press, Bath, UK, 1984.

[PCE$^+$05]   David C. Parkes, Ruggiero Cavallo, Nick Elprin, Adam Juda, Sebastien Lahaie, Benjamin Lubin, Loizos Michael, Jeffrey Shneidman, and Hassan Sultan. ICE: An Iterative Combinatorial Exchange. In *Proceedings of the Sixth ACM Confernece on Electronic Commerce*, 2005.

[PHP$^+$03]   P. Padala, C. Harrison, N. Pelfort, E. Jansen, M. Frank, and C. Chokkareddy. Ocean: The open computation exchange and arbitration network, a market approach to meta computing, 2003.

[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*, pages 1161–1168, 2001.

[PMN05]       S. Parsons, M. Marcinkiewicz, and J. Niu. Everything you wanted to know about double auctions but were afraid to (bid or) ask. Technical report, Department of Computer & Information Science, Brooklyn College, City University of New York, 2005.

[Pre98]       C.. Preist. *Economic Agents for Automated Trading*. Hewlett Packard Laboratories, Bristol, 1998.

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

[PWP$^+$03]   V. S. Pai, L. Wang, K. Park, R. Pang, and L. Peterson. The dark side of the web: An open proxys view. In *Proceedings of the 2nd Workshop on Hot Topics in Networking*, 2003.

[RJB99]     J. Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Model*. Addison-Wesley, Reading, MA, 1999.

[RN98]      Ori Regev and Noam Nisan. The POPCORN Market - An Online Market for Computational Resources. In *Proceedings of the first international conference on Information and computation economies*, pages 148–157. ACM Press, 1998.

[RPH98]     M.H. Rothkopf, A. Pekec, and R.M. Harstad. Computationally Manageable Combinational Auctions. *Management Science*, 44:1131–1147, 1998.

[RS04]      Ali Roumani and David Skillicorn. Large-Scale Resource Selection in Grids. In *Proceedings of the First International Workshop on Grid Computing and its Application to Data Analysis* , 2004.

[RZ94]      Jeffrey S. Rosenschein and Gilad Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. MIT Press, Cambridge, Massachusetts, 1994.

[San96]     T. Sandholm. *Negotiation Among Self-Interested Computationally Limited Agents*. PhD thesis, University of Massachusetts, 1996.

[San99a]    Tuomas Sandholm. An Algorithm for Optimal Winner Determination in Combinatorial Auctions. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 542–547. Morgan Kaufmann Publishers Inc., 1999.

[San99b]    Tuomas Sandholm. *Multiagent Systems: A Modern Introduction to Distributed Artificial Intelligence*, volume 1, chapter 5, Distributed Rational Decision Making, pages 201–258. MIT Press, 1999.

[SC95]      T. W. Sandholm and R. H. Crites. On Multiagent Q-Learning in a Semi-competitive Domain. In *IJCAI-95 Workshop on Adaptation and Learning in Multi-agent Systems*, 1995.

[Set05]     Seti@Home. User statistics. Website, accessed: 12.02.2005, Seti @ Home website (http://setiathome.ssl.berkeley.edu/numusers.html), 2005.

[SF89]      A. Sathi and M. S. Fox. Constraint-directed Negotiation of Resource Allocations. *Distributed Artificial Intelligence 2*, 1989.

[SG89]      K. Sycara and L. Gasser. Multi-agent Compromise via Negotiation. *Distributed Artificial Intelligence 2*, 1989.

[SH05]      M.P. Singh and M.N. Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. John Wiley & Sons, 2005.

[Sho82]     G. Lynn Shostack. How to Design a Service. *European Journal of Marketing*, 16(1):49–63, 1982.

[Sho84]     G. Lynn Shostack. Designing services that deliver. *Harvard Business Review*, 62(1):133–139, 1984.

[Sim57]     H. A. Simon. *Models of Man - Social and Rational*. John Wiley and Sons,, 1957.

[SJ89]      Eberhard E. Scheuning and Eugene M. Johnson. A Proposed Model For New Service Development. *Journal of Services Marketing*, 3(2):25–34, 1989.

[Smi79]     A. Smith. *An inquiry into the nature and causes of the wealth of nations*. Printed for W. Strahan and T. Cadell, 1779.

[Smi56]     Wendell R. Smith. Product Differentiation and Market Segmentation as Alternative Marketing Strategies. *Journal of Marketing*, 21(1):3–8, 1956.

[Smi89]     Veron Smith. Theory, Experiment and Economics. *Journal of Economic Perspective*, 72(5):151–169, 1989.

[SMT02]     Kumaran Subramoniam, Muthucumaru Maheswaran, and Michel Toulouse. Towards a Micro-Economic Model for Resource Allocation in Grid Computing Systems. In *Proceedings of the 2002 IEEE Canadian Conference on Electrical & Computer Engineering*, 2002.

[SNT98]     R. E. Smith and N. N. Taylor. A Framework for Evolutionary Computation in Agent-Based Systems. In *Proceedings of the 1998 International Conference on Intelligent Systems*. ISCA Press, 1998.

[SNVW05a]   B. Schnizler, D. Neumann, D. Veit, and C. Weinhardt. A Multiattribute Combinatorial Exchange for Trading Grid Resources. In *Proceedings of the Research Symposium on Emerging Electronic Markets (RSEEM), Amsterdam, Netherlands*, 2005.

[SNVW05b]   B. Schnizler, D. Neumann, D. Veit, and C. Weinhardt. Trading Grid Services – A Multiattribute Combinatorial Approach. Working Paper, University of Karlsruhe (TH), Germany, 2005.

[SNW04]     Bjoern Schnizler, Dirk Neumann, and Christof Weinhardt. Resource Allocation in Computational Grids - A Market Engineering Approach. In *Proceedings of the WeB 2004, Washington*, pages 19–31, 2004.

[Soc]       Societies of ComputeeS.

[Som01]     Ian Sommerville. *Software Engineering*. Pearson Studium, Muenchen, 6 edition, 2001.

[SP03]      D. Snelling and T. et al. Priol. Next Generation Grid(s). Technical report, European Grid Research 2005 - 2010. Brussels: Information Society - DG, Grids for Complex Problem Solving, 2003.

[SS03]      Mark Satterthwaite and Artyom Shneyerov. Convergence of a Dynamic Matching and Bargaining Market with Two-sided Incomplete Information to Perfect Competition. Technical Report shneyerov-03-12-17-09-36-, Microeconomics.ca Website, December 2003. available at http://ideas.repec.org/p/ubc/pmicro/shneyerov-03-12-17-09-36-43.html.

[SSGL02]    Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. Winner Determination in Combinatorial Auction Generalizations. In *Proceedings of the First International Joint conference on Autonomous Agents and Multiagent Systems*, pages 69–76. ACM Press, 2002.

[SSS00]     Katrina Stanoevska-Slabeva and Beat F. Schmid. Requirements Analysis for Community Supporting Platforms based on the Media Reference Model. *Electronic Markets*, 10(4):250–257, 2000.

[SW93]      Mark A. Sattherthwaite and Steven R. Williams. The Bayesian Theory of the k-Double Auction. In D. Friedman and J.Rust, editor, *The Double Auction Market - Institutions, Theories, and Evidence*, chapter 4, pages 99–123. Addison-Wesley, 1993.

[Tes02]     L. Tesfatsion. Agent-Based Computational Economics: Growing Economies from the Bottom-Up. ISU Economics Working Paper No. 1, Department of Economics, Iowa State University, Ames, February 2002.

[TH93]      D. S. W. Tansley and C. C. Hayball. *Knowledge-based systems analysis and design : a KADS developer's handbook*. Prentice Hall, New York, 1993.

[TRB99]     D. P. Truex and R. R. Baskerville. Growing Systems in Emergent Organizations. *Communications of the ACM*, 42(8):117–123, 1999.

[VB04]      Srikumar Venugopal and Lyle Buyya, Rajkumarand Winton. A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids. Technical Report, GRIDS-TR-2004-1, University of Melbourne, Australia, 2004.

[Vei03]     Daniel Veit. *Matchmaking in Electronic Markets*. Springer Verlag, 2003.

[Vic61]     W. Vickrey. Counter Speculation, Auctions, and Competitive Sealed Tenders. *Journal of Finance*, 16:8–37, 1961.

[Wal54]      L. Walras. *Elements of pure economics*. Allen and Unwin, London, 1954.

[WBPB03]     R. Wolski, J. Brevik, J. S. Plank, and T. Bryan. *Grid resource allocation and control using computational economies*, chapter 32, pages 747–772. Number ISBN: 0-470-85319-0. John Wiley & Sons, 2003. Booktitle: Grid Computing: Making The Global Infrastructure a Reality .

[Wei91]      M. Weiser. The Computer for the Twenty-First Century. *Scientific American*, 256(3):94–104, 1991.

[Wel93]      M. P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelli-gence Research*, 1:1–23, 1993.

[WHH$^+$92]   C. A. Waldspurger, T. Hogg, B.A. Huberman, J. O. Kephart, and W.S. Stornetta. Spawn: A Distributed Computational Economy. *IEEE Transactions on Software Engineering*, 18(2):103–117, 1992.

[WHN03]      Christof Weinhardt, Carsten Holtmann, and Dirk Neumann. Market Engineering. *Wirtschaftsinformatik* , 45(6): 635–640 , 2003.

[WJB03]      R. Wolski and J. J. Brevik. Grid Resource Allocation and Control Using Com-putational Economies. In Berman, F. and Fox, G. and Hey, A., editor, *Grid Computing: Making The Global Infrastructure a Reality*. Wiley & Sons, San Francisco, 2003.

[Wol88]      Asher Wolinsky. Dynamic Markets with Competitive Bidding. *Review of Economic Studies*, 55(1):71–84, 1988. available at http://ideas.repec.org/a/bla/restud/v55y1988i1p71-84.html.

[Wol98]      Rich Wolski. Dynamically forecasting network performance using the Network Weather Service. *Cluster Computing*, 1(1):119–132, 1998.

[WPBB01a]    R. Wolski, J. Plank, J. Brevik, and T. Bryan. G-Commerce: Market Formulations Controlling Resource Allocation on the Computational Grid. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, 2001.

[WPBB01b]    R. Wolski, J.S. Plank, J. Brevik, and T. Bryan. Analyzing Market-Based Resource Allocation Strategies for the Computational Grid. *International Journal of High Performance Computing Applications*, 15(3):258–281 , 2001.

[WW03]       M. P. Wellman and W.E. Walsh. Decentralized supply chain formation: A market protocol and competitive equilibrium analysis. *Journal of Artificial Intel-ligence Research*, 19:513–567, 2003.

[WWW98]      W. Walsh, P. Wurman, and M. Wellman. Flexible double auctions for electronic commerce theory and implementation. *Decision Support Systems*, 24:17–27, 1998.

[WWW01]      Peter R. Wurman, Michael P. Wellman, and William E. Walsh. A Parametrization of the Auction Design Space. *Games and Economic Behavior*, 35(1-2):304–338, 2001.

[WWWMM01] Michael P. Wellman, William E. Walsh, Peter R. Wurman, and Jeffrey K. MacKie-Mason. Auction Protocols for Decentralized Scheduling. *Games and Economic Behavior* , 35:271–303, 2001.

[YB04]       C.S. Yeo and R. Buyya. A taxonomy of market-based resource management systems for utility-driven cluster computing. Technical Report, GRIDS-TR-Grid Computing and Distributed Systems Laboratory, University of Melbourne, 2004.

[Ygg98]      F. Ygge. *Market-Oriented Programming and its Application to Power Load Management*. PhD thesis, Lund University, Sweden, 1998.

[YUWI95]     Hirofumi Yamaki, Kyto University, Michael P. Wellman, and Toru Ishida. A market-based approach to allocating QoS for multimedia applications. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi–Agent Systems*. MIT Press, 1995.

[Zin05]      Florinao Zini. Analysis of Simulation Environment. CATNETS project deliverable D2.1, ITC-irst Trento, Italy, 2005.

[Zmu83]      R.W. Zmud. *Information Systems in Organizations*. Scott, Foresman and Company, Glenview, IL, 1983.

The main content of this report is the identification and definition of market mechanisms for Application Layer Networks (ALNs). On basis of the structured Market Engineering process, the work comprises the identification of requirements which adequate market mechanisms for ALNs have to fulfill. Subsequently, two mechanisms for each, the centralized and the decentralized case are described in this document. These build the theoretical foundation for the work within the following two years of the CATNETS project.