J Grid Computing (2008) 6:325–347
DOI 10.1007/s10723-008-9105-0

A Framework for Commercial Grids—Economic and Technical Challenges

Dirk Neumann · Jochen Stößer · Christof Weinhardt · Jens Nimis

Received: 13 April 2007 / Accepted: 21 May 2008 / Published online: 3 July 2008

Abstract This paper argues that the technology of Grid computing has not yet been adopted in commercial settings due to the lack of viable business models. While in academia Grid technology has already been taken up, the sharing approach among non for-profit organizations is not suitable for enterprises. In this paper, the idea of a Grid market is taken up to overcome this Grid adoption gap. We propose a framework for building up a Grid market and identifies the associated economic and technical challenges. Based on this framework, we identify a catalogue of possible market mechanisms which offer a promising fit to the Grid environment's characteristics and which may thus help to carry the idea of Grid markets from theory to practice.

D. Neumann (✉) · J. Stößer · C. Weinhardt
Chair for Information Systems,
Albert-Ludwigs-Universität Freiburg,
Platz der Alten Synagoge, 79085 Freiburg, Germany
e-mail: dirk.neumann@vwl.uni-freiburg.de

J. Stößer
e-mail: stoesser@iism.uni-karlsruhe.de

C. Weinhardt
e-mail: weinhardt@iism.uni-karlsruhe.de

J. Nimis
FZI Forschungszentrum Informatik,
Haid-und-Neu-Str. 10, 76131 Karlsruhe, Germany
e-mail: nimis@fzi.de

Keywords Dynamic pricing · Electronic markets · Grid computing

1 Introduction

Grid computing denotes a computing model that distributes processing across an administratively and geographically dispersed infrastructure [11]. By connecting many heterogeneous computing resources, virtual computer architectures are created, increasing the utilization of otherwise idle resources. Originally, Grid computing comes from the area of high performance computing where users needed an easy access to massive processing and storage resources. Consequently, the Grid has primarily been used as a capable middleware layer by researchers. In subsequent times, the Grid evolved beyond accessibility towards "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations" [10]. This newly developing view on Grids can be interpreted as a very high-level business model: users of different administrative units share their resources to perform computationally demanding tasks [43].

In academia, this business model has transformed the community into several virtual organizations (e.g. Alice, Atlas, CMS, LHCb), which facilitate resource sharing among its members. This business model seems to work well for

academia where resource sharing is prevalent between those members who contribute resources to the Grid. One of the most prominent activities in academia is the EGEE II project being funded by the European Commission. EGEE brings together researchers from over 27 countries with the common aim of developing a service Grid infrastructure which is suited for any scientific research especially where the time and resources needed for running the applications are considered impractical when using traditional IT infrastructures (e.g. weather forecasts, protein folding, etc). With a funding of over 30 million €, the established EGEE Grid comprises over 20,000 CPUs and 5 Petabytes of storage which are available to scientists 24 × 7. In total, EGEE serves 20,000 concurrent jobs on average. For enterprises, this sharing model has not made it into practice. This is not surprising as administrative barriers–even within companies – are too difficult to overcome to make the sharing approach work. The free-riding problem, implying that members participate in the Grid without contributing [1], coupled with security issues strongly argue against the sharing approach. As such, it is not surprising that the vast potential of Grid computing has not yet been exploited – despite the fact that technical advancements with respect to sensitive issues (such as security and Quality of Service management) have occurred. Nonetheless, even conservative estimates project that Grids may lower total IT costs by 30% [22]. This leads up to the expectation expressed in the report "Grid Computing: A Vertical Market Perspective 2005–2010", which estimates an increase of worldwide Grid spending from $714.9 million in 2005 to approximately $19.2 billion in 2010 [2].

The adoption of Grid Computing by commercial companies has been slow due to the lack viable business models embedded in chargeable Grid services. There is deficit of mechanisms that enable users to discover, negotiate, and pay for the use of Grid services. According to one of the leading Grid research institutes, The451Group, the application of resource trading and allocation models is one of the crucial success factors for establishing commercial Grids [9]. Attaching fair prices to the Grid services in a market-like fashion assures, on the one hand, that resources are only invoked when needed, and, on the other hand, that idle resources are contributed to the Grid.

Complying with the proposition of The451Group, Sun Microsystems has adopted the idea of trading resources within their utility computing initiative, believing that companies will eventually stop maintaining their own infrastructure and instead buy computing power via a Grid. To put weight on this idea, Sun is currently establishing *network.com*,[1] an electronic market-place for trading resources. Sun started out offering a fixed price for computing services of $1 per CPU hour. Amazon is currently launching comparable efforts with Amazon Elastic Compute Cloud (Amazon EC2)[2] and Amazon Simple Storage Service (Amazon S3).[3] Like Sun, Amazon is also offering processing power and storage for a fixed price, but the prices also depend on the consumed bandwidth. Almost every large computer hardware manufacturer like HP or Intel has already worked on or at least pondered the options for Grid markets (cf. [19]). Currently, electronic marketplaces for Grid computing have not yet taken off [34].

This paper attempts to explain why Grid market initiatives have failed so far even in the conception phase of the development. The explanation mainly focuses on the object that is being traded on Grid markets. As a result of this analysis, it is discovered that not only one single Grid market is necessary but a set or catalogue of different marketplaces satisfying the needs of different market segments. This paper provides a framework towards such a catalogue of market mechanisms. As such, this paper provides guidance for potential Grid market operators (e.g. Telecom companies such as France Telecom for Mobile Grids [23], or hardware vendors such as Sun for Computational and Data Grids) in the choice of the market mechanisms that are needed to gear up Grid computing for enterprises.

The remainder of this paper is structured as follows. Section 2 describes why markets work well in Grid environments. Section 3 defines the

[1] http://www.network.com/, 12.11.2007.

[2] http://aws.amazon.com/ec2, 09.11.2007.

[3] http://aws.amazon.com/s3, 09.11.2007.

technical challenges of Grid markets, whereas Section 4 focuses on the associated economic challenges. As a result of Section 4, a two-tiered economic market structure with two classes of markets is proposed as a viable solution for commercial Grids. Section 5 discusses the use of different market mechanisms within this two-tiered market structure and shows which mechanisms are most adequate for which kind of application. Section 6 concludes the paper with a summary and points to future work.

2 Markets for the Grid

2.1 Why Markets

Markets seem to be adequate for commercial Grids, as they offer a business model which charges upon Grid usage. As a side-effect, markets have the ability to improve the resource allocation. Today's resource management systems in Grids have recognized the need of expressing values by including user priority, weighted proportional sharing, and service level agreements that set upper and lower bounds on the resources available to each user or group [13, 21, 25]. Maximizing the utility (i.e. the sum of the users' valuations), however, is only possible if the resource manager knows the attached valuations, meaning the exact relative weights at any point of time. Knowing the valuations at any time is a very demanding requirement, as users typically have no incentive to report decreases in their valuation, because they loose priority and correspondingly value by not getting their computation completed.

Hence, value-oriented approaches are not sufficient per-se to achieve an efficient solution. Only if all participants are willing to report their priorities and values honestly, these algorithms (e.g. Proportional Share [19]) will work well. This is where Grid markets enter the discussion. Markets have the ability to set the right incentives for users to reveal their true valuation as well as for resource owners to provide those resources that are scarcest in the Grid. With the introduction of prices, incentives will be given to the users to substitute the scarce resource, say the number of CPUs, with less scare resource, say memory. For instance, a fixed pricing scheme which requests 10 € for one CPU and 1 € for memory, sets incentives to reduce the number of used CPUs in favor of the cheaper memory.

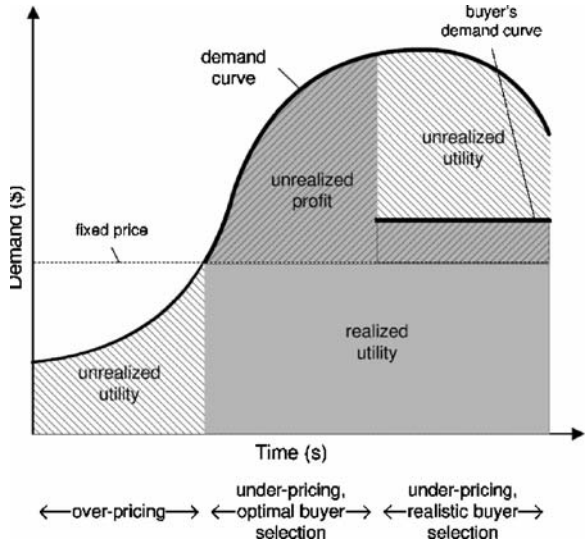

Fig. 1 Fixed pricing [20]

A fixed pricing scheme, however, is not enough to achieve an efficient allocation: Suppose the fixed price of a resource provider as shown in Fig. 1 (cf. [20]). Demand changes over time as depicted by the parabola; without loss of generality the costs of supplying resources are assumed to be zero. If demand is below the fixed price (left end of the graph), no resource will be requested. This is because the value for the resource, represented by the demand curve, is below the price. As a consequence, there is a loss of utility which is shown by the area below the demand curve. In the middle part of the graph there could be more buyers willing to pay the fixed price, as their demand exceeds the price. If the seller allocates the resource efficiently to the user who values the resource most, there will be unrealized profit for the seller. This is indicated by the hatched gray area which refers to the difference between the demand curve and the fixed price. Obviously, market mechanisms work well in scheduling [26].

2.2 Case Study

In this section, we demonstrate the usefulness of market mechanisms in Grids compared to

technical schedulers by referring to a special instance. The example is instructive as it shows how markets work in general.

2.2.1 The Setting

In this case study we consider a Grid environment with a number of individual agents, which are assumed to act rationally in the sense that they strive for maximizing their individual benefit from participating in the Grid. Agents submit computational jobs to the Grid, where a job j is specified by the "type" $t_j = (r_j, p_j, w_j)$. The job's release date is denoted by $r_j \in \mathbb{R}^+$, the job's processing time by $p_j \in \mathbb{R}^+$, and the agent's cost of waiting for one additional unit of time until the job is finished (its "weight") by $w_j \in \mathbb{R}^+$. The quasi-linear utility function of agent j is defined to be $u_j(C_j, \pi_j) = -w_j C_j - \pi_j$, where $C_j \in \mathbb{R}^+, C_j \geq p_j$, marks j's completion time and $\pi_j \in \mathbb{R}^+$ is j's payment [12]. When requesting resources for the execution of job j, the agent may strategically choose to submit the request $\tilde{t}_j = (\tilde{r}_j, \tilde{p}_j, \tilde{w}_j) \neq t_j$ to the Grid. In the following, we consider a decentralized setting where the actions of job j do not only consist of reporting its type $\tilde{t}_j$, but also of subsequently choosing a machine at which to queue. We will henceforth denote agent j's action by s_j, and we will also write $u_j(s, t)$ for agent j's utility given the strategy vector $s = (s_1, \ldots, s_n)$ and the true types $t = (t_1, \ldots, t_n)$. The Grid infrastructure consists of m identical, parallel machines $M = \{1, \ldots, m\}$.

2.2.2 The Mechanism

For this setting, we propose the use of the Decentralized Local Greedy Mechanism (DLGM) [12]. The mechanism aims at minimizing the total weighted completion time $\sum w_j C_j$ across all jobs. DLGM comprises the following three steps:

Step 1 – **Job submission:** At release date $\tilde{r}_j$, j reports $\tilde{w}_j$ and $\tilde{p}_j$ to every machine $m \in M$.

Step 2 – **Real-time planning:** Based on this information, the machines perform real-time planning based on a local scheduling policy. The most frequently used policy follows the "weighted shortest processing time first" principle [36]. Accordingly, jobs are assigned a priority value which depends on their ratio of reported weight and processing time: Job j has a higher priority value than $k \Leftrightarrow \frac{\tilde{w}_j}{\tilde{p}_j} \geq \frac{\tilde{w}_k}{\tilde{p}_k}$. Then j is scheduled before job k in the waiting queue. Depending on the current local waiting queue which resulted from the previous strategies s_{-j} of all other jobs, and j's report $\tilde{t}_j$, machine i reports the tentative completion time $\hat{C}_j(i \mid s_{-j}, \tilde{t}_j)$ to agent j. Formally: Let $H(j)$ be set of jobs with a higher priority than j and $L(j)$ the set of jobs with a lower priority than j. We write $j \to i$ if job j is assigned to machine i. Without loss of generality, jobs are indexed in order of their release dates (i.e. $j < k \Rightarrow \tilde{r}_j \leq \tilde{r}_k$). Finally, let $b_i(l)$ be the remaining processing time of the currently running job on machine i at time l and let S_j denote the point in time when job j is started to be executed. Then $\hat{C}_j(i \mid s_{-j}, \tilde{t}_j)$ can be determined as

$$\hat{C}_j(i \mid s_{-j}, \tilde{t}_j) = \tilde{r}_j + b_i(\tilde{r}_j) + \sum_{j \in H(j), k \to i, k < j, S_k \geq \tilde{r}_j} \tilde{p}_k + \tilde{p}_j,$$

i.e. $\hat{C}_j(i \mid s_{-j}, \tilde{t}_j)$ consists of j's own runtime plus the runtime of the job which is currently being executed on machine i (if any) and the aggregated runtimes of the jobs which would be queuing in front of job j.

Furthermore, the mechanism computes a tentative payment $\hat{\pi}_j(i \mid s_{-j}, \tilde{t}_j)$ according to the prominent Vickrey principle. Job j has to compensate all jobs which are currently waiting at machine i and which are delayed due to j being assigned to i for their additional waiting cost, i.e.

$$\hat{\pi}_j(i \mid s_{-j}, \tilde{t}_j) = \tilde{p}_j \sum_{k \in L(j), k \to i, k < j, S_k \geq \tilde{r}_j} \tilde{w}_k.$$

These values are only tentative as later arriving jobs might displace job j.

Consequently, the tentative utility of job j at machine i at time $\tilde{r}_j$ is

$$\hat{u}_j(i \mid s_{-j}, \tilde{t}_j) = -w_j \hat{C}_j(i \mid s_{-j}, \tilde{t}_j) - \hat{\pi}_j(i \mid s_{-j}, \tilde{t}_j).$$

Step 3 – **Machine selection:** Upon receiving information about its tentative completion time and required payment from the machines, the agent makes a binding decision to queue at a machine i, and pays $\hat{\pi}_j(i \mid s_{-j}, \tilde{t}_j)$ to the delayed jobs.

2.2.3 Data Generation

Our aim is to simulate rational agents with learning capabilities, which may strategically misreport about w_j based on their past behavior and payoffs. We will refer to these strategies as "rational response strategies".

In order to compare the DLGM with a technical scheduler, we ran six settings. To increase the competition in the market, across the course of these settings, we successively increased the Poisson-based arrival rate of jobs from $\lambda = 0.2$ to $\lambda = 1$ as listed in Table 1, which also specifies the random choices of the processing times and weights. Each job sequence was then fed into both the DLGM mechanism with agents playing myopic best response strategies (i.e. agents do not remember the outcomes of earlier market interactions but are somewhat "myopic" in that only consider the current situation). In addition we check whether the DLGM outperforms a technical scheduler in terms of efficiency and incentive compatibility.

For simplification, we assume that agents can only misreport about their weight w_j. This is not too far-fetched, one central result of Heydenreich et al. [12] is that myopic best response agents then truthfully report $\tilde{w}_j = w_j$.

To model the learning capabilities of the agents for the rational response strategy, we chose a reinforcement learning-based approach with a greedy selection policy [15]. This approach com-prises the following two phases:

- **Exploration phase:** In this phase, the agents simultaneously explore the strategy space and heuristically try to learn the expected payoffs of various strategies to finally decide on an expectedly optimal strategy. Within this exploration phase, we perform 25 runs of the market. In each run k, every agent j reports $\tilde{w}_j^k$ and p_j to the market at time r_j and, upon having received the feedback about its tentative completion time and payment from each machine, selects the machine i which maximizes its tentative utility $\hat{u}_j(i \mid s_{-j}, \tilde{t}_j)$, where $\tilde{w}_j^k$ is an integer which is randomly drawn from the uniform distribution with support $[0, 2w_j]$ in order to model both the under- and overstating of weights. Subsequent to each run k, agent j determines its actual ex-post utility (i.e. its "reward") $u_j^k((\tilde{s}_j, s_{-j}), t)$ for round k. After all 25 runs, agent j greedily determines its best strategy s_j^* as $s_j^* = argmax_k u_j^k((\tilde{s}_j, s_{-j}), t)$. Since the strategy space of agent j consists of strategically reporting $\tilde{w}_j$ only, we can simplify this to the greedy optimal strategy being the report of $\tilde{w}_j^*$ which yielded the maximum ex-post utility across all 25 runs.
- **Exploitation phase:** In this phase, the agents try to "exploit" the information which they obtained in the previous exploration phase. Using a greedy selection policy, agent j assumes that, since the report of $\tilde{w}_j^*$ maximized its ex-post utility given its true weight w_j in the past, reporting a fraction of $\frac{\tilde{w}_j^*}{w_j}$ for the weights of all subsequent jobs will also maximize its future payoff. In order to examine this strategy, we also ran the exploitation phase for 25 rounds.

Table 1 Simulation settings

Setting	S1	S2	S3	S4	S5	S6
Arrival time λ of the $\tilde{r}_j$ Poisson-based release dates	0.2	0.5	0.6	0.7	0.8	1
Processing time p_j	Lognormal distribution with mean 9 and variance 3					
Weights w_j	Normal distribution with mean 9 and variance 3					

In each round, we generated one job per agent based on the same distributions as in the exploration phase as specified in Table 1.

2.2.4 Results

Essentially, the mechanism aims at maximizing total utility of the system, which is expressed as minimizing total weighted completion time of all jobs, where the wait defines the opportunity costs of a job incurred by waiting another time unit. The Grid user submits his request to all machines by specifying the release date of the job comprising of its estimated processing time and the weight specifying how expensive it is to delay the job by an additional time unit. The machines compute the tentative completion time of the job on the basis of the reported weight and the payments the job has to make in order to compensate all jobs that are delayed due to the priority of this job. These compensation payments are necessary in order to prevent the users from overstating their weight. The machines report the tentative completion time and payments back to the user who chooses the best available machine.

This mechanism is benchmarked against a naive First-in-First-out technical scheduler. As an evaluation methodology an agent-based simulation with learning agents is used to define the strategies of the job requesters. It is also tested whether the requesters reports weights different from their true weights.

The simulation checks five different settings with varying demand situations, ranging from excessive supply of resources to excessive demand for resources (Table 2). In the learning phase, the job requesters explore the strategy space and learn how to set up their strategy, which boils down to reporting the weight. In the subsequent exploitation phase, several rounds are played with those learned strategies.

The simulation turns out that, when competition is low, technical schedulers achieve as good welfare (i.e. total weighted completion time) as market mechanism does. However, with increasing utilization, technical schedulers are starting to perform worse than market mechanisms. When competition is high, first in, first out generates about 13% higher overall waiting costs than DLGM with agents playing rational response strategies. This result is rather straightforward, as technical schedulers do not account for the relative urgency of jobs. When utilization is low, however, the welfare loss incurred by technical schedulers is insignificant. In addition the revenue gained by users is not supported, as the resources are all equal by assumption. Thus, markets cannot unfold their full potential, as their favorable effects of better scheduling are neglected.

Table 2 Simulation results

Metric	Mechanism \ Setting	S1	S2	S3	S4	S5	S6
Utilization	All mechanisms	38%	77%	91%	100%	100%	100%
Welfare	FIFO	−916,634	−1,993,694	−2,676,904	−3,633,538	−5,046,646	−8,784,801
	DLGM with myopic best response	−916,701	−2,001,682	−2,650,274	−3,340,492	−4,370,345	−7,042,589
	DLGM with learning agents	−916,717	−2,001,787	−2,663,577	−3,485,629	−4,676,194	−7,802,760
Positive compensations	DLGM with myopic best response	1%	7%	16%	18%	23%	34%
	DLGM with learning agents	1%	8%	18%	20%	35%	36%
Lateness	DLGM with myopic best response	0.002	0.073	1.13	8.72	20.73	58.99
	DLGM with learning agents	0.002	0.106	1.198	8.659	19.982	58.79

FIFO first in, first out

This impact of better scheduling is coupled with the participation effect markets exert. As contribution is compensated, enterprises have an incentive to provide idle resource to the Grid. We are not aware of any numerical estimates on the magnitude of this effect, but it is generally thought to be high.

3 Technical Challenges of Grid Markets

Summarizing the discussion above, markets can play a crucial role for commercial Grids as they set the right incentives for contributing resources without over-consumption. In addition, markets improve the scheduling decision over pure technical schedulers. On the other hand, the use of markets need to be integrated with state-of-the-art Grid middleware—otherwise the concept will remain purely theoretic. In this section we present a generic system model for Grid markets to allow for a well-founded discussion of the technical challenges Grid markets raise.

Figure 2 introduces the layered logical architecture of the system model in terms of its functional entities, their responsibilities and their dependencies. The boxes represent functional entities and the arrows indicate dependencies between them, where an arrow from an entity A to an entity B means that entity A depends on entity B, i.e. that A receives input from B or uses B's services.

Layer 4: Grid Applications and Resources Layer 4 comprises the Grid applications as demand, the provided resources as supply and the corresponding users, i.e. the Grid application users and the resource providers. The resource provider makes use of the so-called intelligent tools of layer 3 to model the business strategies and the offered resources. On the consumer side either the user of the Grid application or the Grid application itself will autonomously use the intelligent tools to model the application's resource

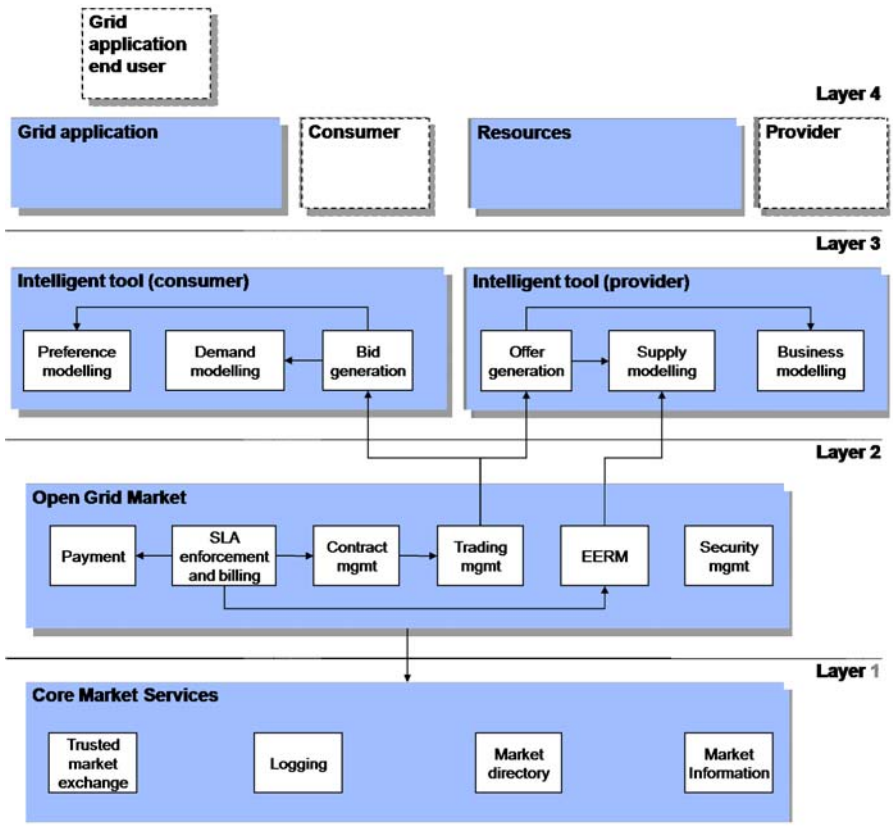

Fig. 2 Layered architecture for Grid markets

requirements based on the user's *economic* preferences. If an application is decomposed into several aggregated services, it is the application's task to realize a service deployment coherent to the allocations made on the market.

Layer 3: Intelligent Tools For an easy access to the Grid market, both the users and the resource providers must be supported by a set of intelligent tools.

- **Demand modeling:** The user needs to be equipped with a tool to specify the technical requirements of his Grid application, i.e. the properties of the resources. In case the requirements are specified in terms of aggregated services, demand modeling also assumes the task of decomposing a request into its constituent services.
- **Preference modeling:** This component facilitates the description of the user's economic preferences that will determine his bidding strategies on the market, e.g. the user can define the maximal amount he is willing to pay or if he prefers cheap over reliable resources.
- **Bid generation:** The bid generation denotes the component that generates and places the bids on the market on behalf of the user. For this purpose the bid generator retrieves the user preferences, the technical requirements and the current state of the Grid market and forms the bid or bid series accordingly.
- **Supply modeling:** The supply modeling component is the provider's correspondent of the user's demand modeling component. This component allows the providers to specify the technical properties of the resources contributed to the Grid (e.g. a Beowulf cluster).
- **Business modeling:** Analogously to the consumer preference modeling, the providers need to specify their business models to generate adequate offers. For example, one part of the description could be a pricing model that specifies if the users have to pay for booked time-slots or for the actual usage.
- **Offer generation:** The offers are derived from the technical supply descriptions and the business model of the respective provider via the offer generation component. As an autonomous agent, the offer generation component continuously observes the Grid market and places and updates offers.

Layer 2: Open Grid Market The second layer is headlined Open Grid Market, as it accounts for the economic matchmaking (i.e. which request is allocated to what resource, when and for what price?). Accordingly, it provides the necessary functionality along the economic matching process, starting from the secure access to the Grid market over the actual allocation to the subsequent billing and payment.

- **Security management:** The security management component serves as entry point for a single sign-on mechanism and is responsible for a tamper-proof identity management for the consumers, the suppliers and the constituent components of the Grid market.
- **Trading management:** The actual matchmaking process that assigns bids to suitable offers is executed by the trading management. As a first step, the trading management matches the technical descriptions of the request (received from the bid generation component) to the technical descriptions of the offered resources (received from the offer generation component). This corresponds to the common task of service discovery and takes into account a potential hierarchical decomposition of the request. In the second phase the trading management orchestrates the bidding process between the users and the providers following the protocol of the employed market mechanism. If the bidding process finishes successfully, the pairs of corresponding bids and offers are submitted to the contract management.
- **Contract management:** The contract management transforms the pairs of bids and offers into mutually agreed contracts. One important part of these contracts are the service level agreements (SLAs) which define the agreed terms of usage of the resources and the pricing. After stating the contract the contract management initiates the SLA enforcement.
- **SLA enforcement and billing:** The enforcement of the SLAs triggers the job submission, keeps track of the resource usage, compares it to the SLA and at the end initiates the billing

and clearing according to the comparison results.

- **Economically Enhanced Resource Management (EERM):** The EERM provides a standardized interface to the resource providing Grid middleware (e.g. Globus Toolkit GT4 or Sun Grid Engine). Comparable with a wrapper, the EERM isolates its clients from middleware specific issues and to enhance and complement its functionality with market relevant features. The EERM's main duties include:
  - Resource fabric management: Standardized interfaces to create instances of resources and later make use of them from the application.
  - Resource management: The resource management aims at the satisfaction of the SLA. It also notifies users and providers of variations and additionally coordinates independent resources to allow for co-allocation in case this is not directly provided by the fabrics.
  - Resource monitoring: The resource monitoring subcomponent monitors the state of the resources in terms of its technical parameters and reports them to the SLA enforcement.
- **Payment:** Once the SLA enforcement and billing component has determined the degree of SLA fulfilment and associated payments, the payment component is invoked. The payment component offers a unified interface to commercial payment services such as PayPal.

Layer 1: Core Market Services Standard Grid middleware does not provide all the infrastructure services necessary for the Open Grid Market. Layer 1 extends the standard Grid middleware by basic infrastructure services:

- **Trusted market exchange service:** All communication among market participants (users, providers and services of the Open Grid Market layer) is mediated by the trusted market exchange service, which assures that information is routed to the appropriate party in a secure and reliable way. Routing is performed by applying rules on content and also from the context of the communication (for instance, existing negotiations).
- **Logging:** All transactions executed on the market must be registered in a secure log for auditing purposes (for instance, to avoid the repudiation of contracts).
- **Market directory:** The market directory is a market-aware extension to the commonplace service registries in standard Grid middleware. The directory contains all currently available resource offers with their technical and economic information and thus provides information about the current market situation as input to the bid and offer generation and to the trading management components.
- **Market information:** The market information service allows participants to publish information and to gather information (e.g. prices, resource usage levels) from other participants. Participants can query the service or subscribe to certain topics.

The implementation of the system model sketched above raises some serious technical challenges that have to be addressed to allow for the vision of an open market for Grid resources. The challenges that are summarized in Table 3 concern market-based Grid research in general, i.e. the technologies and tools used to implement the Grid market.[4]

4 Economic Challenges of Grid Markets

In this section, we will derive the economic challenges associated with building a generic Grid marketplace. We will present a market structure which is centered around the object which is to be traded on the Grid market.

From Market Engineering, it is known that the design of markets ultimately depends on the characteristics of the objects that are being traded over the market. For Grid computing, we

[4] A first prototype has been implemented (cf. [37]).

Table 3 Technical challenges for Grid markets

Challenge	Description
Composition of applications	The dynamic composition of applications from resources that have to be acquired on a market is a specialization of the general problem to compose applications out of existing services. A promising approach to address this issue is the use of semantic technologies—in Grid computing usually denoted as the Semantic Grid.
Standards and stability	The rapid development of current Grid and web service standards leads to permanent changes in the technologies and tools that have to build the basis of the Grid market. This lack of stability in underlying systems makes it hard to mature the Grid market components over time.
SLA formulation and enforcement	Current research has identified more than 600 possible parameters for SLAs from a business perspective. Obviously, it is very complex to negotiate SLAs with such a large numbers of dimensions and to enforce them during execution. Thus, the relevant parameters have to be identified to allow for an appropriate expressiveness of SLAs and to restrict the complexity of its technical management at the same time.
Economic awareness of resources	Current resource fabrics' and Grid middleware's resources are not aware that they are situated in an economic environment, for example they do not know that they have a certain price or that there malfunction implies financial compensation. Thus, resources need extensions to cope with their economic nature and especially to inform clients of their economic as well as technical state.
Transparency	For a good acceptance of Grid markets – besides economic factors – it is also necessary that the developed technology can be used as transparently as possible. The required changes in the existing systems must not be too invasive and the users must not be bothered with too many additional time-consuming tasks. While the users tasks – at least to some extent – can be supported by intelligent agents and wizards, virtualization seems to be a promising remedy against technical complexity.

distinguish three classes of Grid resources based on the level of functionality and the mode of deployment, which depends on how the EERM processes the application:

- *Physical resources* can be CPUs, memory, sensors, other hardware and software or even aggregated resources such as clusters (e.g. a Condor or MOSIX cluster). From a technical point of view, resources are simple to describe as there exists only a finite set of resources. For instance, a resource may be defined by the operating system (e.g. Linux OS), the number of CPUs (e.g. 4 * x86 CPU), memory (e.g. 128MB RAM) etc. The GLUE schema[5] provides a standardized vocabulary for describing computing elements. The standardization of resources offers an easy way to uniquely describe them. This in turn alleviates resource discovery as matchmaking is straightforward.
- *Raw services* are resource-near services that access resources via standardized interfaces. Examples comprise storage Web services and virtualization middleware. Raw services also comprise application services which could

[5] http://forge.ogf.org/sf/projects/glue-wg, 09.11.2007.

potentially be standardized as expressive description languages such as Job Submission Description Language[6] or Resource Specification Language[7] exist.
– Non-standardized application services are *complex application services* (or simply *complex services* in the remainder), which are so diverse that they cannot reasonably be standardized, e.g. services which perform data consolidation and aggregation or statistical analyses. For the description of complex application services domain-dependent ontologies can be used, if existent. Nonetheless, the indefinite search space tremendously exacerbates service description and likewise service discovery of complex services.

Designing one Grid market for all kinds of resources, from physical resources such as processing power, memory and storage running on native platforms to sophisticated virtual resources or services, bundling and enriching such physical resources, seems inappropriate due to both technical and economic factors.

– **Technical factors**: From the technical perspective, differences in the monitoring and deployment of services and resources exist, such that it is very difficult to devise a generic system for the EERM that allows the trading all kinds of resources and services. Even worse, different deployment mechanisms embedded by the EERM impose different requirements on the market mechanism. Physical resources–either accessed directly or as raw services via standardized interfaces – are *application-independent* in the sense that the application is fully transferred to and deployed on the physical resources (given that these resources match the application's technical requirements), but the resources are somewhat generic and can be used for a wide range of applications. Complex services on the other hand offer more specific functionality and can only be used by a limited number of applications. The interface (input and output, e.g. data format and accuracy) of the services needs to exactly match the needs of the calling applications, and complex services are therefore *application-dependent*.
– **Economic factors**: These alternative ways of deployment and dependencies give rise to distinctively different requirements for potential markets. From an economic perspective, application-independent markets (i.e. markets for physical resources and raw services) are promising for automation via an organized electronic market. There are standardized items for sale that potentially attract many buyers and sellers. Complex, application-dependent services have a disadvantage as demand is highly specialized and distributed across niche markets such that only few potential buyers and sellers are interested in the same or a related service. But overall, demand for complex services is huge as most of the potential users are interested in getting their services executed, no matter how many physical resources will be needed. From the economic perspective, market mechanisms need to achieve the standard objectives in mechanism design listed in Table 4. As pointed out above, trading resources and complex services imposes totally different requirements on the market. While resources are more or less a commodity for which auction mechanisms seem to work well, complex services are inherently non-standardized making auction-like mechanisms inapplicable.

From this brief discussion, it can be stated that a one-size-fits-all market mechanism for Grids is infeasible. Instead, due to their heterogeneous properties, Grids can be divided into two different types of markets, *application-independent markets* and *application-dependent markets*, spanning out a "two tiered Grid market structure"[8] (cf. Fig. 3). It should be noted that we can use the same Grid market system model. But we need different instances of the components. In the forefront, the trading management component needs to imple-

[6] http://forge.ogf.org/sf/projects/jsdl-wg/, 09.11.2007.

[7] http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html/, 09.11.2007.

[8] We use the term "market structure" to denote the configuration of marketplaces.

Table 4 Economic objectives

Objective	Description
Allocative efficiency	Allocative efficiency is the overall goal of market mechanisms for Grid resource allocation. A mechanism is said to be allocatively efficient if it maximizes the utility across all participating users (welfare or overall "happiness"), i.e. the sum over the valuations of all winning resource requesters less the sum over the reservation prices of all winning resource providers.
Budget-balance	Budget balance is basically one out of two feasibility constraints. A mechanism is budget-balanced if it does not need to be subsidized by outside payments. The payments coming from the resource requesters cover the payments made to the resource providers.
Individual rationality	Individual rationality is the second feasibility constraint. Individual rationality loosely translates into participation constraint. A mechanism is individually rational if users cannot suffer any loss in utility from participating in the mechanism, i.e. it is individually rational to participate.
Computational tractability	Due to the potentially large number of resource and service requests and offers, the complexity of the underlying allocation problem and the need for immediacy of the allocation decision, the Grid mechanisms needs to be computed in polynomial runtime in the size of its input, i.e. the number of resource requests and offers.
Truthfulness	Truthfulness means that it is a (weakly) dominant strategy for users to reveal their true valuations to the mechanism. Truthfulness is a desirable feature since it tremendously simplifies the strategy space of the users; there is no need to reason about the bidding strategies (even the strategies of the other market participants), reducing the bidding problem to a preference elicitation problem.

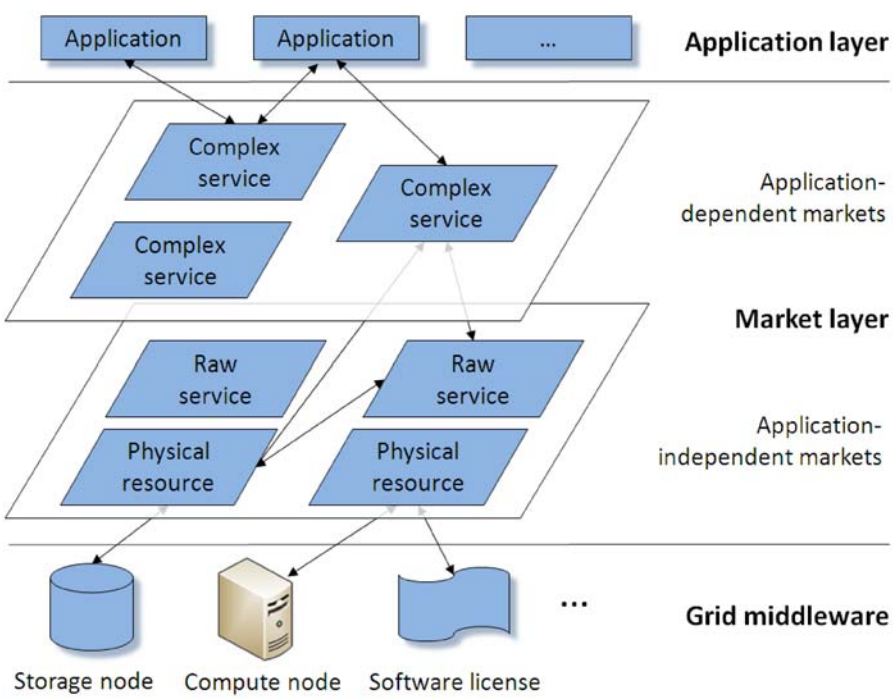

Fig. 3 Two-tiered economic market structure

ment a different market mechanism that supports the respective market. In the following, those two classes of markets are analyzed in terms of their requirements on the market mechanisms.

4.1 Application-independent Markets

On the market for physical resources and raw services, low-level resources are traded such as processing power, memory, and storage. Demand in the application-independent market is generated by complex services that need resources to be executed. This setting in the application-independent market poses special requirements on the used market mechanism [33]:

- *Multi-attributes*: Physical resources, either deployed as raw service or as resource, have quality attributes such as speed of the CPU, the operating platform or bandwidth. Thus the mechanism needs to cope with multi-attributes.
- *Bids on bundles*: Generally, users require a combination of resources to execute a job (e.g. CPU and memory). If the mechanism does not account for bids on bundles, the user is facing the risk of obtaining only one leg of this bundle without the other (the so called "exposure risk"). The market mechanism thus needs to support requests for bundles of resources.
- *Time attributes*: When raw services are traded, the market mechanism needs to take time attributes into account. The requesters need to specify their demand, so that the market mechanism can efficiently schedule the job requests according to availability of resources and to the price. This differs from trading resources, where the market mechanism executes jobs upon availability.
- *Co-allocation*: Capacity-demanding Grid applications usually require the simultaneous allocation of several homogenous service instances from different providers. For example, a large-scale simulation may require several computation services to be completed at one time. This situation with the simultaneous allocation of multiple homogenous services is called co-allocation. A mechanism for the service market has to enable co-allocations and provide functionality to control it.
- *Coupling*: For some applications, it may be logical to couple multiple raw services of a bundle in order to guarantee that these are allocated from the same seller and – more importantly – will be executed on the same machine.
- *Resource isolation*: Security and performance considerations lead to the requirement of resource isolation. In fact, it can only be assured that an application satisfies a given quality-of-service (QoS) or security level, respectively in case the resources on which the application is executed are committed only to one party.
- *Online mechanism*: The allocation of the mechanism needs to be made within seconds. The mechanism thus needs to be a lightweight mechanism that requires little computation time. Being an online mechanisms is crucial because in the case of a decrease in the performance of an application, new resources need to be found and scheduled for immediate execution.
- *Split-proofness*: The mechanism needs to be split-proof in the sense that users cannot improve their priority by splitting jobs into more parts. Consequently, split-proofness can be interpreted as a fairness criterion as small jobs do not get preferred over large jobs.
- *Merge-proofness*: Likewise, the mechanism needs to assure that users do not have advantages through the merger of jobs, i.e. that large jobs do not get preferred over small jobs.

4.2 Application-dependent Markets

On the application-dependent market, applications demand the execution of their constituting complex services. Along the lines of the two-tiered market structure, complex services can be decomposed into smaller raw services that can in turn be translated into resources that are necessary for executing them. E.g., some complex application service might require a basic XML Transformer service which in turn needs processing power, memory, storage etc. Buyers in such a market request a complex service—the provider of this service, the service integrator, is responsible for

obtaining the required raw services and physical resources in turn on the application-independent market, thus hiding parts of the Grid's complexity from the buyer. Such a hierarchical shading of complexity seems to be an appropriate approach since service requesters typically have no information about how much resources the complex service will consume (e.g. [8]).

The requirements on market mechanisms for application-dependent markets are different than for application-independent markets. Complex services are rarely used by two different companies—installing competition in application-dependent markets hence does not make sense. Here the difficulty rather stems from having to find a counterpart that is exactly offering the capabilities to execute the application. The market mechanism is more search-oriented such as bilateral or multi-lateral negotiation protocols, giving rise to the following requirements:

- *Multi-attributes*: Complex services have quality attributes defining the QoS of the service. Thus the mechanism needs to cope with multi-attributes.
- *Workflow support*: To support complex applications, distributed resources such as computational devices, data, and applications need to be orchestrated while managing the application workflow operations within Grid environments. The market-mechanism needs to account for this during design time and run time of the workflow. This imposes extreme difficulties on the market mechanism, as the operations need to be performed in the defined manner opening up lots of exposure risks: if one single task in the workflow fails, the complex services cannot be orchestrated. Thus the market mechanism needs to account for this problem in a rather quick manner.
- *Scalability*: Scalability considers how the properties of a protocol change, as the size of a system (i.e. the participants in the Grid) increases. The market mechanism needs to be scalable per se in order to be applicable.
- *Co-allocation*: Capacity-demanding Grid applications usually require the simultaneous allocation of several homogenous service instances from different providers. For example, a large-scale simulation may require several computation services to be completed at one time. This situation where simultaneous allocation of multiple homogenous services is called co-allocation. A mechanism for the service market has to enable co-allocations and provide functionality to control it.

The problem with current markets for the Grid is that they are purely designed as application-independent markets. For example Sun's $1 advertisement campaign aims at selling physical resources, i.e. CPU hours. This type of market is, however, by design not relevant for enterprise customers who have deadlines to meet until when a job needs to be executed and have no idea about how many physical resources are required to meet the deadline. While enterprises typically have time critical jobs to execute, applications in academia are less time dependent. As such, application-independent markets where the resources (being deployed as resources) would be viable business models – here the users have to wait until the queued jobs are being executed. But clearly, the issue of payment for resources is controversially discussed in academia. In the future, even for the EGEE Grid billing and payment will soon become an issue as demand always exceeds supply. It seems that application-independent markets will become an adequate model for academia Grids such as EGEE or D-Grid.

Grid markets that will be widely accessed by enterprises need to be of the form of application-dependent markets where complex services are offered. For example a manufacturer is interested in executing a Computer Aided Design application and deploying it on a computation intense platform. This complex service showcases a very specific service which is likely to be demanded from a single requester. To accommodate this complex service, the service must be decomposed into its constituting raw services and further on to the physical resource demand. Integrators are needed to facilitate this decomposition process, where integrators denote companies that are specialized in aggregating and disaggregating services into resources. Specialization stems from experience allowing the identification of the service

demand needs by comparing it with similar services in the past, where similarity is established in terms of algorithms, data structures and sizes etc. Telecommunication companies and hardware producers seeking to virtualize IT infrastructures naturally have a strong interest and the competency to become integrators.

It should be noted that for complex services that are offered by one designated firm to multiple customers (e.g. clone ERP such as SAP's hosted solution customer relationship management service), the concept of software-as-a-service (SaaS) gains traction. Those complex services are typically offered by fixed prices and do not require a Grid market.

5 A Catalogue of Dynamic Pricing Mechanisms

In the previous section, we have motivated that Grids require not something like a Global Grid market where all Grid requests and supplies are collected, but a more complex two-tiered market structure. Figure 3 summarizes the two-tiered market structure. On application-independent markets, complex services require physical resources either plainly deployed or accessed via service interface. Applications demand the execution of several complex services on application-dependent markets. Integrators assume the responsibility of mediating between the applications being unaware of their resource need and the resources themselves.

In the following we will set up a taxonomy of known market mechanisms which supports different types of Grid applications. This taxonomy is conceived to be a roadmap for further Grid market developments that help bridging the adoption gap.

Obviously, the requirements on the market mechanism stem, firstly, from the definition of the trading object, which in turn determines how applications are deployed and, secondly, from the application mode. Dividing the market structure into two parts – application-independent and application-dependent markets – is too simplistic, as the timing when demand occurs has not yet been considered. This timing is determined by the application itself and depends on the task the application is performing. We use the term application mode as a characterization of the processing mode of the application. This encompasses in particular the workload of the application as well as the interaction model between applications and the Grid middleware virtualizing the execution platform. Depending on the application mode, different requirements upon the market mechanisms emerge.

- **Batch applications** are characterized by a planned execution and expected termination time. Execution is serial and resource demand depends on parameters such as the size of the input data. Essentially, most Grid applications are batch applications; jobs can be distributed to nodes that have idle resources. The results are reported back. Examples are compute-intense applications like data mining or distributed search algorithms.
- **Interactive applications** are those applications that require services or resources on demand, depending on the interactions with users. Different than batch applications, with interactive applications it is not possible to plan execution and expected termination time far in advance, so there can be unpredictable peaks of requests occurring within a short time. Examples for interactive applications are online data analyses such as what-if scenarios.
- **Task-oriented applications** are dynamically composed of single tasks to build more complex tasks. Service demand depends on the (work-)flow of requests from multiple users. For example the transaction system of a bank constitutes a task-oriented application. From a resource allocation point of view, task-oriented applications are extremely demanding, as the resource demand follows a workflow of tasks. Exposure problems – getting one service or resource without the others – are prevalent. Thus the allocation mechanism needs to consider the whole set of tasks as partial allocations can be inefficient.

Most of the market-based approaches relate to batch applications. Batch applications are comparably easy for two main reasons. Firstly, there is no need to consider a whole workflow with different resource demands on each echelon of

the workflow. Secondly, the time to determine the allocation can be relatively long; immediacy is not essential. Thus, complex resource allocation computations can be performed without hampering the whole application due to latency times devoted to the calculation of the optimal allocation. Even worse, most of the practical market-based Grid prototypes consider only one single resource type (e.g. CPU only) and thus make use of standard auctions such as the English or Dutch auction [30, 42]. For applications other than pure number crunching, those auction types are inadequate as more than one object (e.g. memory, broadband, and operation system) is required at the same time.

In the following, we will position existing market mechanisms within our two-tiered market structure as well as regards these application modes.

5.1 Mechanisms for Application-independent Markets

5.1.1 Batch Applications

As mentioned above, the market mechanisms for raw services and physical resources depend on the application mode. In the following we discuss the mechanisms that are adequate for batch applications.

- **Multi-attribute Combinatorial Auction** (Bapna et al. [4]): In this market mechanism, multiple requesters and providers can trade both computing power and memory for a sequence of time slots. First, Bapna et al. introduce an exact mechanism for solving the multi-attribute combinatorial auction problem. By introducing so-called fairness constraints and imposing one common valuation across all resource providers, they structure the search space in the underlying combinatorial allocation problem as to establish one common, truthful price per time slot for all accepted requests. Additionally, this introduces a linear ordering across all jobs and time which reduces the complexity of the allocation problem, which however still remains NP-hard. To mitigate this computational complexity, they thus propose a fast, greedy heuristic at the expense of both truthfulness and efficiency. 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 all orders are aggregated to one virtual bid - this is more appropriate for Cluster computing, where all resources are equal and under control of one entity. Moreover, the mechanism does not take co-allocation constraints into account.
- **Multi-attribute Exchange** (Stößer et al. [39]): In the spirit of the mechanism developed by Bapna et al. [4], Stößer et al. [39] propose a heuristic for clearing Grid markets in order to overcome the computational intractability of exact mechanisms. The heuristic is designed so as to obtain an approximately efficient allocation schedule at low computational cost while accounting for strategic, self-interested users in a heterogeneous environment. In contrast to Bapna et al.'s mechanism, it preserves truthfulness on the request-side of the market while accounting for heterogeneous resource and service providers under decentralized control. Nonetheless, coupling, co-allocation and resource allocation are currently not supported by this mechanism.
- **Multi-Attribute Combinatorial Exchange (MACE)-mechanism** (Schnizler et al. [33]): In MACE users are allowed to request and offer arbitrary bundles of grid services for which they can specify QoS-attributes and coupling constraints. MACE implements an exact mechanism for solving a combinatorial, multi-attribute scheduling problem. The scheduling problem in this combinatorial setting is NP-hard, where complexity dramatically increases with the number of bids being introduced. This paper introduces a new pricing scheme for combinatorial exchanges, the k-pricing rule. In essence, the k-pricing rule determines the price such that the resulting surpluses to the buyers and sellers divide the entire surplus being accrued by the trade according to the ration k. The k-pricing rule is budget-balanced but cannot attain the efficiency

property. As simulations have shown, the k-pricing rule introduces an incentive to report valuations at least approximately truthfully. For batch applications, where the allocation can be defined way before the execution time, MACE appears to be a valid alternative for operating batch raw service markets.

- **Combinatorial Scheduling Exchange** (AuYoung et al. [3]): A combinatorial scheduling exchange can also work for batch applications, where the language allows the specification of bundles. AuYoung et al. propose an exact combinatorial exchange for computing resources, where the pricing is based on the approximation of the truthful VCG prices proposed by Parkes et al. [28]. Exact combinatorial problems require a lot of computation time and become infeasible once the problem size is rising. Thus, the combinatorial scheduling exchange can only be used for batch applications. The mechanism does not account for co-allocation, coupling, and resource isolation constraints.
- **Augmented Proportional Share** (Stoica et al. [41]): In the initial version the proportional share mechanism works without time constraints. For trading services this is not impossible; Proportional share has been augmented by time attributes. Depending on the amount of the bid for certain time slots, the allocation is done according to the bids. Augmented proportional share is a very straightforward but very limited mechanism, as only one standardized good can be allocated. Although time constraints can be managed, all other requirements (i.e. multi-attributes, bids on bundles, co-allocation, coupling, resource isolation) are not supported.

5.1.2 Interactive Applications

For interactive applications it is impossible to predict demand for raw services and resources. Thus, the mechanisms need to allocate continuously. This can be realized by either frequent call mechanisms, where bids are collected for a very short time span and cleared right away. This requires that the mechanism is solvable in few seconds. Alternatively, the mechanism could be an online mechanism, which allows the real-time job submission to available resources (e.g. nodes or clusters). A third way is to introduce a derivative market to insure against the risk of supernormal resource demand.

- **Fair Share**: Essentially, the group of fair share mechanisms (e.g. example being SHARE [16]) are technical schedulers and not market mechanisms, as resource assignments are not based on prices bids. We list it here, despite this fact, due to their easy implementation and importance in practice. Different to other technical schedulers the idea behind fair share is equal treatment of all users rather than focusing on processes. Fair share denotes a scheduling strategy in which the usage of a certain resource (either CPUs or number of nodes in a cluster) is equally distributed among system users. Such a scheduling mechanism is inadequate for Grids as it gives more resources to users with more processes. Thus, it is not split-proof, meaning that a user can indefinitely increase its share by splitting the processes into several smaller ones.[9]
  Variants of the fair share mechanism have been introduced that allow the administrator to partition users into groups and apply the fair share principle to these groups as well. The most common way of implementing the fair share scheduling strategy is to apply a recursive round-robin strategy. The drawback of the fair share strategy is that all parameters are pre-specified and set by the system administrator. The buyers have no influence on the allocation. From an economic perspective fair share strategy reaches only a very low level of efficiency; except in the case that all buyers have the same utility for a share of the resource. Fair share can be applied to aggregated resources (e.g. nodes) allowing the introduction of bundles.
- **Proportional Share** (Chun and Culler [6], Lai et al. [19]): In recent years proportional share schedulers have made their way into practice. Essentially, each user submits a bid that is

[9] On the other hand, fair share is merge-proof as no participant can gain a higher share by merging processes.

used to determine the share of the resources the user is allocated with. The higher the bid, the higher is the share the users will have to expect. Proportional share thus epitomizes a market system in its purest form. Especially in overload situations, proportional share has the desirable property of being more flexible keeping the share of resources constant, where the resource amount is dwindling down. There are many variants of proportional share out. For example, in many Grid systems the weights attached to users are kept constant, leaving the share constant without paying attention to the actual demand situation. Proportional share has the desirable property of being merge-proof but is not split-proof [24]. Typically proportional share is used in association with only one resource, ignoring the use of bundles.

– **Pay-as-bid** (Sanghavi and Hajek [31]): The pay-as-bid mechanism proposed by Sanghavi and Hajek extends the pay-as-bid mechanisms of Kelly [17] and Johari and Tsitsiklis [14] by introducing a discriminatory instead of uniform pricing. Discriminative prices are designed to remove the effect of uniform prices where high bidders tend to bid less than their true valuation to avoid an increase in the prices per share. In pay-as-bid mechanisms the buyers submit only a single real value as bids i.e. the willingness-to-pay. These bids equal the final payment each buyer has to contribute. The allocation of the good is generated according to a pre-specified allocation mechanism. A mechanism that is optimal for two buyers is presented by Sanghavi and Hajek in [32]. They show that the proposed mechanism leads to a unique Nash Equilibrium and furthermore is the most efficient mechanism when pure price bids are being used. The mechanism reaches a worst case fractional efficiency of 87.5%. For the extension for n buyers the worst case efficiency can no longer be kept upright. Only a lower bound can be determined for the worst case efficiency, which is still close to theoretical maximum ($87.03\% < \text{efficiency} < 87.5\%$). Apparently, the mechanism can only guarantee that the outcome is close to the maximum [32]. The pay-as-bid mechanism exceeds proportional share in terms of efficiency. On the other hand pay-as-bid does not provide bid on bundles.

All three mechanisms, however, have the same drawback, as they can be used in scenarios where one resource provider serves several consumers. In all cases there is no competition among the providers. In case, all resources are under centralized control this is unproblematic, but in Grids, it is the idea to cross administrative boundaries, and mechanisms are needed which allow for multiple providers.

– **Multi-attribute Combinatorial Auction Heuristic** (Bapna et al. [4]): As for batch applications, the multi-attribute combinatorial auction can be used for continuous resource allocation processes as well. In this case, it is necessary to rely on the heuristic, since otherwise the problem is too computation intensive. Obviously, the same shortcomings for batch applications also apply for interactive ones.
– **Multi-attribute Exchange** (Stößer et al. [39]): Again, the multi-attribute exchange mechanism can also be applied here, retaining the shortcomings as for batch applications, though.
– **DLGM** (Heydenreich et al. [12], cf. Section 2.2): All the previous mechanisms are applicable due to their computational tractability; however, they still require the periodical clearing instead of supporting (near) real-time allocation decisions. To this end, Heydenreich, Müller et al. [12] propose a DLGM which aims at maximizing the users' overall "happiness" by minimizing the sum of the jobs' weighted completion times. The mechanism is taken from the general machine scheduling domain and extended with a pricing scheme which guarantees budget-balance. It does, however, not achieve full incentive compatibility with respect to jobs reporting their true characteristics, neither does it sufficiently account for the inter-organizational character of Grids. Machines are assumed to be under centralized control. Stößer et al. [40] extend the mechanism so

that those drawbacks in Grid settings are alleviated.

- **Augmented Proportional Share** (Stoica et al. [41]): This mechanism can also be used for interactive applications, while the same restrictions apply as before.
- **Derivative Markets** (Kenyon and Cheliotis [18]; Rasmusson [29]): Whereas the mechanisms introduced before are spot markets being concerned with the immediate trade of commodities, derivative markets trade contracts that specify rights or obligations to receive or deliver payments based on future events. Derivatives yield two benefits: allowing, firstly, to hedge against risk, and secondly, to speculate.
  It is a well known result from finance that it is possible to setup a portfolio consisting of options and the underlying assets that is both risk-free and self-financing, i.e. the investor does neither need to add nor remove capital from the portfolio in order to perform the hedge [29]. This also enhances capacity planning, as futures and options generate early signals about future prices which may help to better control the system by inducing users to distribute excess demand over time or provide incentives for service providers to contribute to the Grid system [38]. Derivative markets can be used to price network resources. A certain fraction of network capacity is represented by a so called "capacity share". A capacity share is traded on a spot-market and contains the non-expiring right to use the underlying network capacity, i.e. the holder of a share is guaranteed to be able to use the corresponding share of the network for an indefinite period of time. Holders of such a share who do not need this capacity anymore are required to sell their share on the spot-market. By trading options on these capacity shares, users can hedge against risk: The holder of a capacity share can hedge against the risk of not being able to sell this share in the future by entering into a so called "put option"; A user who needs the guarantee to be able to use some network capacity in the future can reserve this capacity by buying a so called "call option". As pointed out above, options cannot only be used for hedging purposes but also for speculation (arbitrage) which contributes liquidity to the market.

5.1.3 Task-oriented Applications

The requirements on market mechanisms that support task-oriented applications are very demanding, as all constituents of the workflow needs to be allocated – otherwise the application has no value to the user. Currently, there are only bargaining protocols available that guides the user in its search for all components of the workflow.

- **Bargaining Protocol** (Czajkowski et al. [7]): The bargaining involves multiple rounds between the users (e.g., co-allocators) and providers (e.g., schedulers) until an agreement is negotiated (e.g. Siddiqui et al. [35]). The providers post their offers on request to the user, who can either select among the available offers or enter re-negotiation by relaxing some constraints.

5.2 Mechanisms for Application-dependent Markets

As aforementioned trading complex services is very demanding, as there are not many providers and requesters existing. Currently, there is not much research available that aims at developing market mechanisms for trading complex services. Hence, the market mechanisms for batch, interactive and task-oriented applications do not differ substantially.

5.2.1 Batch Applications

For batch applications there are three different market mechanisms suitable, where the first one, MACE, provides a sufficiently rich bidding language for supporting complex services. The second mechanism is the bargaining protocol, while the last one refers to take-it-or-leave-it pricing, where the vendor sets the price and the users decide whether or not to purchase.

- **MACE-mechanism** (Schnizler et al. [33]): Albeit, MACE can represent complex services, it does not support workflows. In ad-

dition, the scalability requirement is not fully satisfied.
- **Bargaining Protocol** (Czajkowski et al. [7]): Essentially, bargaining protocols are the only mechanisms that can be used for trading complex services.
- **SaaS**: SaaS refers more to model of software delivery where a service provider (e.g. SAP) offers to requesters applications that are specifically implemented for one-to-many hosting. This on-demand software selling is coupled with fixed take-it-or-leave-it prices. The risk of peak loads is shifted to the software provider. It should be noted that SaaS applies to somewhat standardized applications allowing the implementation for one-to-many hosting.

5.2.2 Interactive Applications

Interactive applications make the design of adequate market mechanisms even more difficult. Potentially, the following mechanisms could be used.

- **Bargaining Protocol** (Czajkowski et al. [7])
- **SaaS**
- **DLGM** (Heydenreich et al. [12]): This online scheduling mechanism could also be used for complex services. The problem with this mechanism is that it does not support co-allocation of resources.

5.2.3 Task-oriented Applications

As aforementioned, task-oriented applications are very demanding due to the exposure risks involved. Mechanisms that could be used for trading are bargaining protocols and SaaS. It should be pointed out that there is currently almost no research in this field available—a notable example is the work of Blau et al [5].

- **Bargaining Protocol** (Czajkowski et al. [7])
- **SaaS**
- **Path Auction** (Blau et al. [5]): Blau et al. propose an auctioning protocol that supports workflows. Essentially, each service requester invites different providers of constituting services to the path auction. Every service provider submits a bid depending on the invoking service. This way, it is possible to account for different usage patterns of the service. The payments are determined dependent on the prices without this service provider present. If the price for the complex application service – calculated as the sum of payments for the single service – exceeds the willingness-to-pay, the auction fails. This is intended to confine the auction prices of the service requester to his maximum willingness to pay. In essence, this path auction boils down to an instance of a Vickrey-Clarke-Groves mechanism.

6 Summary

This paper argues that the technology of Grid computing has not yet been adopted by enterprises due to the lack of viable business models. In academia, Grid technology has already been taken up, but the sharing approach among non for-profit organizations cannot be transferred to enterprises. We pick up the idea of a Grid market to overcome this adoption gap. This idea is not new by any means, but hitherto all proposals had been made by computer scientists being unaware of economic algorithms and models or by economists being unaware of the technical possibilities. This paper attempts to derive an economically sound set of market mechanisms based on a solid understanding of the technical possibilities.

Section 2 motivated the need for commercial Grids based on dynamic market mechanisms.

In Section 3, we presented a system model of the Grid market with its functional components and dependencies. The implementation of this architecture raises several technical challenges, such as instable "standards" and SLA formulation and enforcement.

Section 4 analyzed the economic requirements. The nature of the trading object is closely associated with the deployment of software applications. Deployment as resource or as service has major ramifications on the trading object and consequently on the requirements on market mechanisms. Resources are essentially commodities, where services can be both standardized com-

Table 5 Overview of market mechanisms for Grids

Application mode	Application-independent markets	Application-dependent markets
Batch	Multi-attribute Combinatorial Auction [4], Multi-attribute Exchange [39], MACE [33], Combinatorial Scheduling Exchange [3], Augmented Proportional Share [41]	MACE [33], Bargaining Protocol [7], SaaS
Interactive	Fair Share [16], Proportional Share [6, 19], Pay-as-Bid [31], Multi-attribute Combinatorial Auction Heuristic [4], Multi-attribute Exchange [39], DLGM [12], Augmented Proportional Share [41], Derivative Markets [18, 29]	Bargaining Protocol [7], SaaS, DLGM [12]
Task-oriented	Bargaining Protocol [7]	Bargaining Protocol [7], SaaS, Path Auction [5]

modities (i.e. raw services) and non-standardized unique entities (i.e. complex services).

Based on the subsequent analysis, this paper derived a two-tiered market structure where the markets on each tier demand for different market mechanisms. The first tier comprises the application-independent markets for physical resources (e.g. CPU, memory) that can be accessed either as resource or as raw service. The second tier comprises the application-dependent markets for complex services.

Section 5 showed which mechanisms are suitable for what area. Table 5 summarizes the main contribution of this paper—the identification of a set of existing market mechanisms that can be used depending on the application and the respective tier within the market structure. As shown in the paper, for some application types adequate market mechanisms exist, while for others no mechanisms have been identified yet.

In essence, the results of this paper suggest several intriguing research avenues:

- Analyze the properties of the proposed mechanisms for the respective application mode classes.
- Compare the efficiency of the market mechanisms attributed to the different classes of the market mechanism canon.
- Implement the mechanisms and conduct field studies in order to get real data.
- Develop market mechanisms, where the market mechanism canon is mostly silent (e.g. task-oriented applications).
- Develop sustainable business models for companies that provide market platforms for trading Grid services or resources.
- Identify the size and the potential revenue of the single markets of the two-tiered market structure.
- Identify limits of the use of market mechanisms in Grid.

Acknowledgements This work has been partially funded by the EU IST programme under grant 034286 "SORMA–Self-Organizing ICT Resource Management", and by the German D-Grid initiative under grant 01|G07008B "Biz2Grid".

References

1. Adar, E., Huberman, B.A.: Free riding on Gnutella. First Monday **5**(10), 134–139 (2000)
2. Anonymous: Grid Computing: A Vertical Market Perspective 2005–2010. The Insight Research Corporation, Boonton (2005)
3. AuYoung, A., Chun, B.N., Snoeren, A.C., Vahdat, A.: 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, Boston, 9–13 October 2004
4. Bapna, R., Das, S., Garfinkel, R., Stallaert, J.: A market design for Grid computing. INFORMS J. Comput. **20**, 100–111 (2008)
5. Blau, B., Lamparter, S., Neumann, D., Weinhardt, C.: Planning and Pricing of Service Mashups. In: IEEE Joint Conference on E-Commerce Technology (CEC'08) and Enterprise Computing, E-Commerce and E-Services (EEE '08), Washington, D.C., 21–24 July 2008

6. Chun, B.N., Culler, D.E.: Market-based Proportional Resource Sharing for Clusters. Computer Science Division, University of California, Berkeley (2000)
7. Czajkowski, K., Foster, I., Kesselman, C., Sander, V., Tuecke, S.: SNAP: A protocol for negotiating service level agreements and coordinating resource management in distributed systems. In: Proceedings of the 8th Workshop on Job Scheduling Strategies for Parallel Processing, Edinburgh, 24 July 2002
8. Eymann, T., Neumann, D., Reinicke, M., Schnizler, B., Streitberger, W., Veit, D.: On the design of a two-tiered Grid market structure. In: Proceedings of the Multikonferenz Wirtschaftsinformatik (MWKI), Passau, 20–22 February 2006
9. Fellows, W., Wallage S., et al.: Grid Computing—The State of the Market. The451Group, New York (2007)
10. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the Grid: enabling scalable virtual organizations. Int. J. High Perform. Comput. Appl. **15**(3), 200 (2001)
11. Foster, I., Kesselman, C.: The Grid 2. San Francisco, Morgan Kaufmann (2003)
12. Heydenreich, B., Müller, R., Uetz, M.: Decentralization and mechanism design for online machine scheduling. Technical Report, METEOR, Maastricht research school of Economics of TEchnology and ORganizations (2006)
13. Irwin, D.E., Grit, L.E., Chase, J.S.: Balancing risk and reward in a market-based task service. In: Proceedings of the 13th International Symposium on High Performance Distributed Computing (HPDC13), pp. 160–169. IEEE, Piscataway (2004)
14. Johari, R., Tsitsiklis, J.: Efficiency loss in a network resource allocation game. Math. Oper. Res. **29**(3), 407–435 (2004)
15. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. J. Artif. Intell. Res. **4**, 237–285 (1996)
16. Kay, J., Lauder, P.: A fair share scheduler. Commun. ACM **31**(1), 44–55 (1988)
17. Kelly, F.: Charging and rate control for elastic traffic. Eur. Trans. Telecommun. **8**, 33–37 (1997)
18. Kenyon, C., Cheliotis, G.: Forward price dynamics and option design for network commodities. In: Proceedings of the Bachelier Finance Society 2^{nd} World Conference, Knossos, Crete, 12–15 June 2002
19. Lai, K., Rasmusson, L., Adar, E., Zhang, L., Huberman, B.A.: Tycoon: an implementation of a distributed, market-based resource allocation system. Multiagent Grid Syst. **1**(3), 169–182 (2005)
20. Lai, K.: Markets are dead, long live markets. ACM SIGecom Exchanges **5**(4), 1–10 (2005)
21. LSF: http://www.platform.com (2007). Accessed 12 November 2007
22. Minoli, D.: A Networking Approach to Grid Computing. Wiley, Hoboken (2005)
23. Mossmann, M., Stößer, J., Neumann, D., Savourey, D., Krisnashwamy, R.: A Combinatorial Exchange for Complex Grid Services. Working Paper (2007)
24. Moulin, H.: On scheduling fees to prevent merging, splitting and transferring of jobs. Math. Oper. Res. **2**, 266–283 (2007)
25. Neumann, D., Lamparter, S., Schnizler, S.: Automated bidding for trading Grid services. In: Proceedings of the European Conference on Information Systems (ECIS), Gothenburg, 12–14 June 2006
26. Neumann, D., Veit, D., Weinhardt, C.: Grid economics: market mechanisms for Grid markets. In: Barth, Th., Schüll, A. (eds.) Grid Computing: Konzepte, TechnologienAnwendungen, pp. 64–83. Vieweg Verlag, Wiesbaden (2006) (German)
27. Neumann, D., Borissov, N., Stößer, J., See, S.: Best myopic vs. rational response: an evaluation of an Online Scheduling Mechanism 70. Wissenschaftliche Jahrestagung des Verbands der Hochschullehrer für Betriebswirtschaft e.V. 2008, Berlin (2008)
28. Parkes, D.C., Kalagnanam, J., Eso, M.: Achieving budget-balance with Vickrey-based payment schemes in combinatorial exchanges. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI), Seattle, 4–10 August 2001
29. Rasmusson, L.: Network capacity sharing with QoS as a financial derivative pricing problem: algorithms and network design. Doctoral Dissertation, Royal Institute of Technology, Stockholm (2002)
30. Regev, O., Nisan, N.: POPCORN market. Online markets for computational resources. Decis. Support Syst. **28**(1), 177–189 (2000)
31. Sanghavi, S., Hajek, B.: Optimal allocation of a divisible good to strategic buyers. In: Proceedings of the 43rd IEEE Conference on Decision and Control-CDC, Atlantis, Paradise Island, Bahamas, 14–17 December 2004
32. Sanghavi, S., Hajek, B.: A new mechanism for the free-rider problem. In: Proceedings of the ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems, Philadelphia, 22 August 2005
33. Schnizler, B., Neumann, D., Veit, D., Weinhardt, C.: Trading Grid services – a multi-attribute combinatorial approach. Eur. J. Oper. Res. **187**(3), 943–961 (2008)
34. Shneidman, J., Ng, C., Parkes, D., AuYoung, A., Snoeren, A.C., Vahdat, A., Chun, B.N.: Why markets could (but don't currently) solve resource allocation problems in systems. In: Proceedings of the 10th USENIX Workshop on Hot Topics in Operating Systems, Santa Fe, 12–15 June 2005
35. Siddiqui, M., Villazon, A., Fahringer, T.: Grid capacity planning with negotiation-based advance reservation for optimized QoS. In: Proceedings of International Conference for High Performance Computing, Networking and Storage (SuperComputing), SC06, Tampa, 11–17 November 2006
36. Smith, W.E.: Various optimizers for single-stage production. Naval Resour. Logist. Quart. **3**, 59–66 (1956)
37. SORMA consortium: Preliminary specification and design documentation of the SORMA components. Deliverable D2.1 of the EU FP6 project 034286 "SORMA–Self-Organizing ICT Resource Management" (2007)
38. Spinler, S., Huchzermeier, A., Kleindorfer, P.R.: The Valuation of Options on Capacity. Working

Paper, WHU, Otto-Beisheim Graduate School of Management, and University of Pennsylvania, The Wharton School (2002)
39. Stößer, J., Neumann, D., Anandasivam, A.: A truthful heuristic for efficient scheduling in network-centric Grid OS. In: Proceedings of the 15th European Conference on Information Systems (ECIS), St. Gallen, 7–9 June 2007
40. Stößer, J., Roessle, C., Neumann, D.: Decentralized online resource allocation for dynamic web service applications. In: Proceedings of the IEEE Joint Conference on E-Commerce Technology (CEC'07) and Enterprise Computing, E-Commerce and E-Services (EEE'07), Tokyo, 23–26 July 2007
41. Stoica, I., Abdel-Wahab, H., Jeffay, K.: On the duality between resource reservation and proportional share resource allocation. In: Multimedia Computing and Networking Proceedings, SPIE Proceedings Series vol. 3020, pp. 207–214. SPIE, Bellingham (1997)
42. Waldspurger, C.A., Hogg, T., Huberman, B.A., Kephart, J.O., Stornetta, W.S.: Spawn: a distributed computational economy. IEEE Trans. Softw. Eng. **18**(2), 103–117 (1992)
43. Weishaeupl, T., Donno, F., Schikuta, E., Stockinger, H., Wanek H.: Business in the Grid: the BIG project. In: Grid Economics & Business Models (GECON 2005) of Global Grid Forum 13 (GGF13), Seoul, March 2005