A Decentralized Auction Framework to Promote Efficient Resource Allocation in Open Computational Grids

Laura Kang
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
kang@eecs.harvard.edu

David C. Parkes
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
parkes@eecs.harvard.edu

ABSTRACT

Computational grids enable the sharing, aggregation, and selection of (geographically distributed) computational resources and can be used for solving large scale and data intensive computing applications. Computational grids are an appealing target application for market-based resource allocation especially given the attention in recent years to "virtual organizations" and policy requirements. In this paper, we present a framework for truthful, decentralized, dynamic auctions in computational grids. Rather than a fully-specified auction, we propose an open, extensible framework that is sufficient to promote simple, truthful bidding by end-users while supporting distributed and autonomous control by resource owners. Our auction framework incorporates resource prediction in enabling an expressive language for end-users, and highlights the role of *infrastructure* in enforcing rules that balance the goal of simplicity for end users with autonomy for resource owners. The technical analysis leverages simplifying assumptions of "uniform failure" and "threshold-reliability" beliefs.

1. INTRODUCTION

Computational grids enable the sharing, aggregation, and selection of (possibly geographically distributed) computational resources, and can be used for solving large-scale and data intensive computing applications. Distributed ownership and users with competing needs make market mechanisms, where prices coordinate decision making both within and between organizations, a good fit for solving the resource allocation problems in grids. Market mechanisms promote efficiency: in the short run, resources are allocated to the best use, and in the long run, prices provide signals to promote good long-term investments in appropriate resources. Market mechanisms also enable *policy*, and together with appropriate macroeconomic controls may offer *the* compelling solution to the problem of "virtual organizations" that has taxed the grid community [7].

Substantial advances have been made in grid software and its conceptual framework in recent years. However, real-world open science grids are still limited in size to hundreds of sites and thousands of computers. The complexity in expressing a user's needs coupled with the lack of flexibility, optimization and autonomy in the allocation of resources have been some of the main obstacles. *Users should be able to focus on computational experiments, not gaming to obtain sufficient resources, and organizations should be able to share resources in a flexible and locally managed manner.*

In a companion paper [3], we describe our vision for market-based computational grids and the broad scope of the ***Egg*** project. Egg is a collaborative project between Harvard and Boston University, involving high-energy physicists, computer scientists, and economists.

Here we provide additional details about the *microeconomic* aspect of Egg and ignore macroeconomic issues (e.g. currency supply, inflation, questions of policy, etc.). These are of equal importance and part of the fabric of Egg, components of which are currently being implemented and will be deployed as a practical platform.

The auction framework is designed around the following principles:

- **Efficiency:** in the long-run, as the system adapts, high value jobs should be allocated to resource owners best able to run the job at low cost (subject to other policy constraints), and low value jobs should not be scheduled in an under-provisioned gird.
- **Simplicity:** A simple bidding language is provided for end-users and the framework is *strategyproof* for users, and thus non-manipulable.[1] Strategyproof mechanisms mean that users need not engage in wasteful counter-speculation about how best to game the system and can focus on *using* grids rather than gaming grids.
- **Decentralized control**: Resource owners retain control and flexibility over their own resources. A complete auction (e.g. with pricing policy) is not specified, but rather the framework provides a minimal set of rules that ensure truthfulness.
- **Extensibility:** Resource owners can replace and improve components (e.g. pricing strategies, resource estimation algorithms) over time. This supports innovation.[2]

[1]Given the simplifying assumptions of uniform-failure and threshold-reliability beliefs, to be discussed in Section 2.2.
[2]See Clark et al. [5] for a related, informative discussion of the role of extensibility in the success of the Internet.

We provide no details about *how* resource owners can implement algorithms for resource estimation or pricing in this paper. Instead we shall focus on the design of the auction framework. The novelty here is that we consider a dynamic model in which jobs arrive over time and allocation decisions need to be made dynamically. These dynamics complicate the question of strategyproofness (introducing new opportunities for manipulation). They also make the problem a poor fit for existing technologies, such as combinatorial auctions [6] and combinatorial exchanges, that assume all bids are present at the same time. This does, however, fit nicely into the current trend towards dynamic mechanisms in computational mechanism design [14].

1.1 Related Work

A number of existing systems employ market-based approaches to allocate computational resources [15, 13, 2, 1, 4]. Gomoluch and Schroeder [9] compare auctions with conventional methods. However, none of these papers consider dynamic manipulations, none provide the bid expressiveness that is supported here, and none embrace the need for open, flexible and decentralized control in computational grids. Galstyan et al. [8] share our motivation for a minimalist infrastructure where resource owners can dynamically define their own models but without advocating the use of market-based methods. Several recent papers have focused on dynamic auctions. See Parkes [14] for a recent survey. Our model can be seen as an extension of Hajiaghayi et al. [10] that allows for general value schedules, additional parameters (e.g., job description and reliability metric) and supports strategyproofness via a minimal set of rules.

2. MODEL AND KEY COMPONENTS

We consider the problem of allocating computational resources on compute servers when jobs can have different size/lengths and arrive dynamically. A similar problem of allocating storage space and time can be defined for file servers, but is not analyzed in this paper. We do not study combinatorial issues related to how a user structures her work across multiple jobs (see Section 3.2); rather we assume that each user has a predetermined set of jobs.

In overview, a user can bid for resources at any time by describing her job and annotating the job with a *value schedule* that defines her willingness to pay as a function of the job completion time. This bid spawns a *one-time reverse auction* in which qualified resource providers compete for the right to execute the job. The market infrastructure completes this auction by consulting *price tables* maintained by each resource provider (representing the bids of the resource providers) and the *resource estimates* of each resource provider for the job. The auctioneers (and not the sellers) look up a price to quote to the user in the context of the reverse auction after performing payoff-maximization on behalf of the user. The market infrastructure enforces rules on price tables and resource estimation that ensure strategyproofness for end users.

2.1 Actors

Our model consists of four types of actors: the market infrastructure, auctioneers, users, and resource providers:

Market infrastructure: Trusted by both users and resource providers and implemented as part of the Egg system.

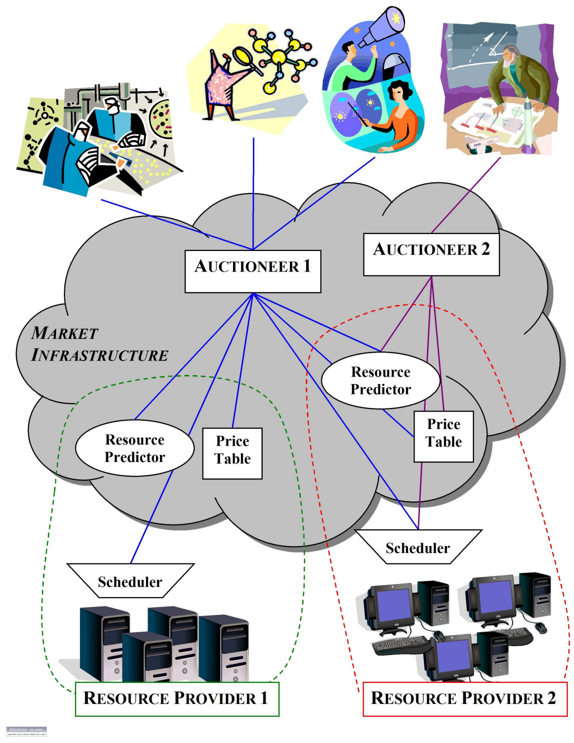

Figure 1: Actors

One important role is to impose constraints on the resource providers (e.g. on the price tables, resource estimates) in order to ensure strategyproofness. The infrastructure also maintains and publishes statistics which the auctioneers can use to compute the reliability metrics for resource providers.

Auctioneers: Part of the infrastructure, a "single-use" auctioneer is created on-the-fly for a user whenever she runs a job. This auctioneer is personalized in terms of the collection of resource providers from which the auctioneer should solicit offers (i.e., the user's own view of the grid.)

Users: Individuals or organizations who wish to employ the resources available on the grid to run jobs, e.g., a physicist who wants to run Monte Carlo simulations.

Resource Providers: Resource providers maintain a collection of computational resources and schedule and perform jobs that are won through auctions. Each resource provider maintains a resource prediction module and price tables within the market infrastructure, and also runs its own local scheduler.

2.2 Modeling Users: Jobs and Beliefs

Each job has a description, an arrival time, and a user has a value schedule for the completion time of the job, and a "minimum reliability threshold" for resource providers from which she wishes to solicit offers. The arrival time of a job is the earliest time the value of the job is known to the user, and the job can be described. A user can be associated with multiple jobs. Taken together, type, $\theta_i = (J_i, w_i, a_i, \gamma_i)$, completely characterizes the information about a job:

- J_i is a string describing the job to be submitted. J_i can be thought of as a list of fields for executable files, input files, number of loops, etc., and specifies how to perform the job (e.g., file location). The default syntax and semantics are defined by the market infrastructure. We assume that attributes can be extracted from J_i, to be used for resource prediction.

- $w_i : [a_i, a_i + \Delta] \rightarrow \mathcal{R}_{\geq 0}$ is a function of the *completion time* where $w_i(\tau)$ is a nonnegative real number representing the willingness to pay for job completed by time τ. We call w_i the value schedule for job i. We assume every job has *bounded patience*, i.e., there exists a minimal constant $\Delta \in \mathcal{R}_{\geq 0}$ such that $w_i(\tau) = 0$ for all $\tau > a_i + \Delta$, for all jobs i. We assume Δ is known to the market infrastructure and resource providers.[3]

- $a_i \in \mathcal{R}_{\geq 0}$ is the arrival time

- $\gamma_i \in [0, 1]$ is the minimum tolerable reliability on resource providers that is acceptable to the job.

Throughout this paper, let d_i denote the latest time τ for which $w_i(\tau) > 0$, i.e., the maximal deadline or departure time of the job. We assume that a user has no value for receiving a completed job outside of a job's arrival-departure interval, or for an incomplete job.

The *reliability metric* for each resource provider is a measure of the frequency with which the provider has successfully completed a job by the scheduled completion time. Note that while reliability is a similar concept to reputation, we emphasize that reliability here is defined objectively from information available to the market infrastructure and maintained by the infrastructure.

A *bid* for a job consists of the job description, value schedule and an *optional* parameter specifying a *minimal reliability.* The arrival time constrains when a user can bid but is not explicitly included in a bid. For example, a bid with a linear value schedule could define ("download Atlas 5.x"[4],(10, 2,Apr-01-06 00:00:01), 99%), where the second component describes a monotonically decreasing willingness-to-pay of $(2 + (10 - 2)(t_d - \tau))/(t_d - t_0)$, where τ is the time of completion, t_d is the maximal deadline (Apr-01-06 00:00:01 in the example), and t_0 is the time at which the bid is submitted to the auctioneer. The auctioneer should then only accept offers from resource providers with reliability $\geq 99\%$. A simple special case is a constant willingness-to-pay with a hard deadline.

We assume that there is a partial order $\succ$ on job descriptions such that $J' \succ J$ whenever a job with description J' has as much value as a job with description J, to a user who has a job with description J. In other words, we assume that if $J' \succ J$, then any user with a job with description J values another job with description J' as much. For example, a job with description $J' \succ J$ may run on a larger input file, request more Monte Carlo iterations, or use a more recent software installation version number (assuming backwards compatibility). Given this we formalize what it means for a user to be "single-minded" in our model:

DEFINITION 1 (SINGLE-MINDED). *Let J_i be the description for job i. Define $\nu_i(J'_i, \tau)$ to be the value to the user (who submits job i) if she is given some completed job with description J'_i by time τ. A user is single-minded if:*

$$\nu_i(J'_i, \tau) = \begin{cases} w_i(\tau), & \text{if } J'_i \succ J_i; \\ 0, & \text{otherwise.} \end{cases}$$

This assumption of single-minded users with respect to some partial order $\succ$ may be restrictive in some settings. This is because it must uniformly hold for all users; e.g., a more recent software version must either be backwards compatible for everyone or backwards compatible for no-one. But the assumption is useful because it allows the strategyproofness of our auction framework to be established without requiring accurate resource predictions (see Section 2.3).

Our technical results about strategyproofness rely critically on two simplifying assumptions about user beliefs. *Threshold-reliability* beliefs are beliefs that a user holds about the comparative reliability across resource providers (and for a particular job) while *uniform-failure* beliefs are beliefs that a user holds about the comparative reliability across different jobs (and for a particular provider). Denote the set of resource providers with reliability metric at least γ_i by Γ_i.

DEFINITION 2 (THRESHOLD-RELIABILITY BELIEFS). *A user has threshold-reliability beliefs for job i with description J_i if she holds belief that all resource providers $\in \Gamma_i$ are equally likely to successfully complete a specific job with description J'_i such that $J'_i \succeq J_i$, and that success is independent of the completion time.*

By this assumption, a user does not reason about the probability that the winning resource provider will successfully complete a specific job. For example, if $\gamma_i = 90\%$, the user is indifferent between a resource provider with reliability 93% and another resource provider with reliability 95% (given that both yield the same payoffs at their respective scheduled completion times).

DEFINITION 3 (UNIFORM-FAILURE BELIEFS). *A user has uniform failure beliefs if she holds beliefs that $Pr($ A job with description J'_i successfully completes if scheduled with resource provider $k) \leq Pr($ A job with description J_i successfully completes if scheduled with resource provider $k)$, for all jobs with description $J'_i \succ J_i$, for all k with reliability at least γ_i.*

With uniform-failure beliefs, a user believes that she cannot improve the probability that a job completes by replacing it with another job with description $J'_i \succ J_i$. Note that the uniform failure assumption is about completion probabilities *if scheduled by the same resource provider.* It allows for different providers to have different levels of reliability, but requires that this does not vary for any one provider when conditioned on the job being accepted by that provider.

2.3 Resource Prediction: Auction Rules

A resource predictor takes the description J_i and the arrival time a_i of job i as inputs, and outputs a vector Q_i of estimated resource requirements. A resource provider can use any learning algorithm it chooses, but the market infrastructure imposes a monotonicity requirement that constrains the estimates for different jobs and also how the model can be refined across time. We assume that the learned model has a concise representation so it can easily be verified by the

[3]It is easy to extend our framework to allow for different Δ values for different times of the day, or for users of different kinds of resources.

[4]http://atlas.web.cern.ch/Atlas/index.html

infrastructure (e.g., naive Bayes, continuous Gaussian linear regression, decision tree, etc.).

A resource provider can also select periods in which it is *inactive*, where it does not generate any resource estimates (but simply queues jobs arriving in this period such that an estimate will be given once the inactive period is over, if these jobs are still not scheduled by any other resource provider).

Let $Q_{ik}^t = \hat{R}_k^t(J_i) \in (\mathcal{R}_{\geq 0})^L$ denote the L-dimensional vector of resource requirements given J_i produced by the learned model used by resource provider k to estimate resource requirements at time t.

Definition 4 (job description monotonicity). *Resource provider k's estimator $\hat{R}_k^t(J)$ is monotonic if $\hat{R}_k^t(J') \geq \hat{R}_k^t(J)$ for all $J' \succ J$, for all t.*

Monotonicity with respect to the partial order on jobs is a reasonable property. For example, using a larger input file or a running a larger number of Monte Carlo iterations of a loop should not require less disk space or runtime. In another sense it is *without loss*, in that if the job is explicitly annotated with enough information and some $J' \succ J$ requires less resources than J, then a resource provider can perform J' for the user instead, and set $R_k^t(J) = R_k^t(J')$. For instance, installing Atlas 5.x should never have a lower resource estimate than installing Atlas 4.x because if Atlas 5.x actually requires less resources (and is backwards compatible, as implied by 5.x $\succ$ 4.x), then whenever Atlas 4.x needs to be installed, Atlas 5.x can be installed instead.

Monotonicity of the resource estimates with respect to *time* is also imposed, to ensure that users cannot benefit from misreporting the arrival time, i.e., delaying job submission, hoping to get a lower price from a lower future resource estimate. Recall that Δ defines the maximal user patience. Time monotonicity is defined as:

Definition 5 (time monotonicity). *Resource provider k's estimator $\hat{R}_k^t(J)$ is monotonic with respect to time if $\hat{R}_k^{t'}(J) \geq \hat{R}^t(J)\ \forall\ J$, for all $t < t' < t + \Delta$ such that resource provider k is not inactive in either t and t'.*

If a resource provider wishes to introduce a new model with lower estimates for some jobs then it can become *inactive* for a period of length at least Δ. No inactive period is required if the new model does not decrease resource estimates.

2.4 Price Tables: Auction Rules

A resource provider maintains a *price table* visible to the market infrastructure, and a function of a vector of resource requirements Q and the completion time τ. It can be thought of as a table of dimension $1+dim(Q)$ where $dim(Q)$ is the dimension of the vector Q and the last dimension corresponds to period in which the job is to be completed. The price table is used by the market infrastructure to generate a resource provider's bid, given a reverse auction for a particular job in some time period.

The ability of resource providers to fill the price table based on their own objectives is a key feature that provides autonomous control. A resource provider can update its price table entries to incorporate scheduling constraints, meet a target load level, or to improve its revenue. However, in order to maintain strategyproofness for end-users, the market infrastructure imposes that the price table entries are *admissible.*

Let $\phi_k^t(Q, \tau)$ denote the price table entries quoted by resource provider k in period t for resource requirement Q for the completion time τ. $\phi_k^t(Q, \tau)$ can be interpreted as: the price (quoted in time t) that resource provider k wishes to receive for completing a job with resource requirements Q, if it were scheduled to complete by time τ. Given a job, a *scheduled completion time* τ_k^* for each resource provider k is selected by the auctioneer, such that the payoff of the user is maximized.

Let $Q[l]$ denote the lth component of the vector Q, and let Δ be the bound on user patience.

Definition 6 (admissibility). *Price table entries $\phi_k^t(Q, \tau)$ are admissible if both:*

$$\phi_k^t(Q'[l], \tau) \geq \phi_k^t(Q[l], \tau) \quad \forall Q'[l] > Q[l], \forall l, \forall \tau, \forall t \qquad (1)$$

$$\phi_k^{t'}(Q, \tau) \geq \phi_k^t(Q, \tau) \quad \forall t' > t, \forall t' \leq \tau \leq t + \Delta, \forall Q. \qquad (2)$$

Prices are admissible if (1) the price table entries are non-decreasing in each component of the resource requirements, e.g., if Q consists of runtime (r) and disk space (s)

$$\phi_k^t((r', s), \tau) \geq \phi_k^t((r, s), \tau)\ \forall r' > r$$
$$\phi_k^t((r, s'), \tau) \geq \phi_k^t((r, s), \tau)\ \forall s' > s$$

and (2) the price table entry for a given completion time within the user-patience window $(t + \Delta)$ does not decrease over time. Note that this still allows a resource provider to decrease prices outside of the user-patience window.

Example 1 (Admissible prices). *Consider the following 2-dimensional price table where each row corresponds to estimated memory usage, and each column to the scheduled completion time. Let t_0 denote the current time. Suppose the intervals for memory are $[0, 2)$, $[2, 4)$, $[4, 8)$ MegaBytes, and the intervals for a job's scheduled completion time is $[t_0, t_0 + 1)$, $[t_0 + 1, t_0 + 2)$, $[t_0 + 2, t_0 + 3)$,$[t_0 + 3, t_0 + 4)$ hours. Suppose the table entries are currently:*

memory	$\tau \in [0,1)$	$\tau \in [1,2)$	$\tau \in [2,3)$	$\tau \in [3,4)$
[0,2)	*5*	*4*	*6*	*10*
[2,4)	*17*	*13*	*15*	*12*
[4,8)	*20*	*15*	*21*	*25*

Suppose $\Delta = 3$ hours. Then, by admissibility the resource provider can increase or decrease all the entries beyond the user patience window. For example, the entries 10, 12, and 25 from the fourth column of the table can be updated to 4, 16, and 18, respectively. However, the other entries cannot be decreased. An updated price table with admissible entries may look like:

memory	$\tau \in [0,1)$	$\tau \in [1,2)$	$\tau \in [2,3)$	$\tau \in [3,4)$
[0,2)	*8*	*4*	*7*	*4*
[2,4)	*18*	*17*	*15*	*16*
[4,8)	*24*	*18*	*100*	*18*

Note that earlier completion times may have higher prices. While we might expect later completion times to have lower prices that earlier ones, the system does not require this.

3. AUCTION FRAMEWORK

We now define the rules of the reverse auction that is created on-the-fly each time a user submits a job. As well as defining the auction, which determines which resource provider (if any) gets to run the job the Egg auction framework imposes the following additional requirements:

- When a user reports job $\hat{J}_i$ via her bid, then this describes the job that will be performed (a user cannot report $\hat{J}_i$ and then have the resource perform some other job, J_i, instead).

- A user makes no payment and does not receive the results from an incomplete job if the job is not completed by the scheduled completion time.

Both of these are requirements are reasonable properties to police by the market infrastructure. Consider the following auction protocol:

1. On receiving bid $(\hat{J}_i, \hat{a}_i, \hat{w}_i, \hat{\gamma}_i)$:
   (a) Let $\hat{d}_i$ be the latest time τ such that $\hat{w}_i(\tau) > 0$.
   (b) Consider the resource providers with reliability at least $\hat{\gamma}_i$. Ensure the resource provider does not change its estimator function based on information about the job.
      i. For each resource provider k, compute the resource estimates $\hat{R}_k^{\hat{a}_i}(\hat{J}_i)$. Let $\hat{r}_k$ denote the estimated runtime.
      ii. The relevant price table entries are revealed to the auctioneer by the market infrastructure. Given the price table entries at time $t = \hat{a}_i$, the auctioneer computes τ_k^* where τ_k^* is the earliest time in $[\hat{a}_i + \hat{r}_k, \hat{d}_i]$ such that:
      $$\tau_k^* = \arg \max_{\tau \in [\hat{a}_i + \hat{r}_k, \hat{d}_i]} \{\hat{w}_i(\tau) - \phi_k^{\hat{a}_i}(\hat{R}_k^{\hat{a}_i}(\hat{J}_i), \tau)\} \quad (3)$$
   (c) Select a resource provider k^* with maximal utility. Denote the set of resource providers with reliability at least $\hat{\gamma}_i$ by $\hat{\Gamma}_i$. Let k^* be
   $$k^* = \arg \max_{k \in \hat{\Gamma}_i} \{\hat{w}_i(\tau_k^*) - \phi_k^{\hat{a}_i}(\hat{R}_k^{\hat{a}_i}(\hat{J}_i), \tau_k^*)\} \quad (4)$$
   The price that a user submitting job i whose type is $\hat{\theta}_i = (\hat{J}_i, \hat{a}_i, \hat{w}_i, \hat{\gamma}_i)$ faces is:
   $$p_i(\hat{J}_i, \hat{a}_i, \hat{w}_i, \hat{\gamma}_i) = \phi_{k^*}^{\hat{a}_i}(\hat{R}_{k^*}^{\hat{a}_i}(\hat{J}_i), \tau_{k^*}^*) \quad (5)$$
   The job is not scheduled if all resource providers have prices higher than value. Break ties in favor of earlier times.
   (d) Collect payment from job i and place in escrow.
   (e) The market infrastructure requires that resource provider k^* encrypts the outcome of job to prevent job i from accessing outcome until $\tau_{k^*}^*$.
2. On the scheduled completion time $\tau_{k^*}^*$:
   (a) Check whether job is completed. Update resource provider reliability.
   (b) If completed, allow job to transfer outcome of computation.
   (c) If completed, transfer payment from escrow to resource provider. Else return payment to job.

A resource provider can employ a scheduling algorithm of choice, and retains autonomy to decide when to actually schedule a job. Notice though, that to get paid for a job it wins, it must schedule the job so the job can be completed on or before the *scheduled completion time* τ_k^*, specified by the auctioneer.

Example 2 (Looking up prices in a price table). *Given a job request and a price table then a row is chosen based on the estimated resource requirement of the job. The ultimate price is determined based on the optimal scheduled completion time, which is chosen by the reverse auction to be where* value - price *is maximal among the entries before the deadline, breaking ties earlier.*

memory	$\tau \in [0,1)$	$\tau \in [1,2)$	$\tau \in [2,3)$	$\tau \in [3,4)$
[0,2)	*9*	*5*	*4*	*6*
[2,4)	*19*	*17*	*13*	*15*
[4,8)	*24*	*20*	*15*	*21*

Suppose the estimated memory usage is 2.5 megabytes and the value schedule given by the user is $w_i([t_0, t_0+2)) = 19$, $w_i([t_0+2, t_0+3)) = 17$, and $w_i([t_0+3, t_0+\Delta]) = 0$. Then the auctioneer examines the row 19, 17, 13, and chooses the scheduled completion time of 2 hours from now, at a price of 13 (i.e., the entry corresponding to $[t_0+2, t_0+3)$), since 17-13 = 4 is the maximum value - price among the corresponding entries.

3.1 Simplicity for Users

Given that users are modeled with *threshold-reliability* beliefs we work with the following relaxed notion of strategyproofness:

Definition 7 (t-strategyproofness). *An online mechanism with limited misreports (no early arrivals) is t-strategyproof if all users hold threshold-reliability beliefs, and no user has incentive to misreport her job description, value schedule, arrival time, or minimum tolerable reliability, regardless of the reports of other users.*

A formal definition of t-strategyproofness is provided in the longer version of this paper [11]. A t-strategyproof mechanism chooses the most favorable resource provider for a user with threshold-reliability beliefs when given the user's true parameters J_i, w_i, a_i, and γ_i, for any reports of the other users.

Theorem 1. *The auction framework is t-strategyproof for jobs with bounded patience and users with limited misreports, uniform-failure and threshold-reliability beliefs, if price table entries are admissible, resource predictors satisfy monotonicity for each resource provider.*

Proof. Omitted. See longer version of the paper. □

We give some intuition for this result. By monotonicity of the resource estimator and admissible prices, a user cannot receive a lower price by reporting a later arrival time or $J_i' \succ J_i$. (Note that the resource estimate does not need to be accurate for the auction to remain truthful.) The uniform failure belief assumption implies that for a specific job J_i, a user has no incentive to report $J_i' \succ J_i$ to achieve a higher probability of success on a specific resource provider. Threshold-reliability beliefs spare us from reasoning about the probability that a job will complete on the winning resource provider.

3.2 Discussion: Strategic Properties

First, we discuss properties that are essential for the non-manipulability of the grid market:

- The market infrastructure requires that the resource provider holds the result of a job until the scheduled completion time. This is to prevent a job from benefiting by overstating its patience and getting a lower price, while still getting the result of the computation early enough.
- The market infrastructure ensures that the price quotes do not change based on the bid of a job. It is a role of the auctioneer, not of the resource providers, to select the best

scheduled completion time for a user and perform the payoff-maximization decision. Resource providers define admissible prices but can update prices only in between receiving bids.

• Prices are set based on *scheduled completion times* rather than based on the actual times in which jobs are performed. This is to prevent a resource provider from deliberately rescheduling the job in order to extract more revenue.

• Completion risk is carried by resource providers, in that if they fail to complete a job by the scheduled completion time, they receive no payment and the user makes no payment (in this case, the user also receives no benefit). This prevents users from benefiting from misreports of their value schedules.

One possible form of useful manipulation that remains, e.g. when prices are super-additive in the size of resource allocations, is for a user to split a job into multiple smaller jobs. This kind of manipulation has been observed in a deployed market for sensor-network resources in the presence of very strategic users [12]. On balance, we have chosen not to preclude this because we believe that super-additive prices can provide opportunities for significant improvement in overall social welfare, e.g. by price discrimination with "rich" users with large jobs that are unable, or unwilling, to split up jobs into smaller jobs having the potential to subsidize the usage of smaller users.

One additional concern that is relevant to the performance of the system is what happens when resource estimates are poor quality, as would be expected when a new user enters the system or when a new class of jobs are run. Here, we can augment our proposal to allow a user to *optionally* state explicit (computational) resource requirements.[5] A user should exercise this "override option" and report explicit information about resource requirements, either to get a lower price by providing a lower estimate or to ensure a successful completion by providing a higher estimate. If a user believes that all resource providers have accurate estimators, then she has no incentive to override the estimates.

4. CONCLUSIONS

We presented a framework with which to support truthful, dynamic and decentralized auctions in computational grids. Our framework is designed to be extensible, aligned with incentives, simple for users, and allows for distributed and autonomous control of resources. We deliberately strive for a strategyproof, simple interface for users (our mantra is "physicists just want to run their jobs") while assuming that resource providers are more willing to compete through sophisticated pricing algorithms and other innovations. In a sufficiently competitive grid this competition should propel improvements in overall efficiency, for instance through better and better resource predictors and schedulers. We are building an initial version of the Egg system, part of which will encompass the microeconomic framework outlined in this paper.

[5]The auction would also need to be changed slightly, so that the resource estimate is adopted as a hard limit on the amount of resources provided to a job. This prevents a new manipulation in which a user under-reports resource requirements of a job but is able to complete her work anyway for a lower price. This change was not needed for the current model because of the coupling of resource estimation with job descriptions.

Acknowledgments

We would like to thank the EGG team members, John Brunelle, Peter Hurst, Chaki Ng, Margo Seltzer, and Saul Youssef for helpful discussions. This work is partially supported by NSF ITR 0427348.

5. REFERENCES

[1] D. Abramson, R. Buyya, and J. Giddy. A computational economy for grid computing and its implementation in the Nimrod-G resource broker. *Future Gener. Comput. Syst.*, 18(8):1061–1074, 2002.

[2] M. Backschat, A. Pfaffinger, and C. Zenger. Economic-based dynamic load distribution in large workstation networks. In *Euro-Par, Vol. II*, pages 631–634, 1996.

[3] J. Brunelle, P. Hurst, J. Huth, L. Kang, C. Ng, D. Parkes, M. Seltzer, J. Shank, and S. Youssef. Egg: An extensible and economics-inspired open grid computing platform. In *Proceedings of the 2006 Grid Asia*, Singapore, May 2006.

[4] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic models for resource management and scheduling in grid computing. *Concurrency and Computation: Practice and Experience (CCPE)*, 14(13-15):1507–1542, 2002.

[5] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden. Tussle in cyberspace: Defining Tomorrow's Internet. In *Proc. SIGCOMM'02*, 2002.

[6] P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. MIT Press, January 2006.

[7] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. of SuperComputer Applications*, 15(3), 2001.

[8] A. Galstyan, K. Czajkowski, and K. Lerman. Resource allocation in the grid using reinforcement learning. In *Proc. AAMAS*, 2004.

[9] J. Gomoluch and M. Schroeder. Market-based resource allocation for grid computing: A model and simulation. In *Proceedings of the First International Workshop on Middleware for Grid Computing (MGC '03)*, 2003.

[10] M. T. Hajiaghayi, R. Kleinberg, M. Mahdian, and D. C. Parkes. Online auctions with re-usable goods. In *Proc. ACM Conf. on Electronic Commerce*, pages 165–174, 2005.

[11] L. Kang and D. C. Parkes. A decentralized auction framework to promote efficient resource allocation in open computational grids. Technical report, Harvard University, 2007.

[12] C. Ng, P. Buonadonna, B. N. Chun, A. C. Snoeren, and A. Vahdat. Addressing Strategic Behavior in a Deployed Microeconomic Resource. In *Proc. 3rd Workshop on Economics of Peer to Peer Systems*, 2005.

[13] N. Nisan, S. London, O. Regev, and N. Camiel. Globally distributed computation over the internet - the POPCORN project. In *ICDCS '98: Proceedings of the The 18th International Conference on Distributed Computing Systems*, page 592, Washington, DC, USA, 1998. IEEE Computer Society.

[14] D. C. Parkes. Online mechanisms. In N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, chapter 16. Cambridge University Press, 2007.

[15] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta. Spawn: A distributed computational economy. *Software Engineering*, 18(2):103–117, 1992.