Algorithmic Mechanism Design for Load Balancing in Distributed Systems *

Daniel Grosu   and   Anthony T. Chronopoulos
Department of Computer Science,
University of Texas at San Antonio,
6900 N. Loop 1604 West, San Antonio, TX 78249
{dgrosu, atc}@cs.utsa.edu

Abstract

Computational Grids are large scale computing system composed of geographically distributed resources (computers, storage etc.) owned by self interested agents or organizations. These agents may manipulate the resource allocation algorithm in their own benefit and their selfish behavior may lead to severe performance degradation and poor efficiency. In this paper we investigate the problem of designing protocols for resource allocation involving selfish agents. Solving this kind of problems is the object of mechanism design theory. Using this theory we design a truthful mechanism for solving the static load balancing problem in heterogeneous distributed systems. We prove that using the optimal allocation algorithm the output function admits a truthful payment scheme satisfying voluntary participation. We derive a protocol that implements our mechanism and present experiments to show its effectiveness.

1. Introduction

In current distributed systems such as computational grids, resources belong to different self interested agents or organizations. These agents may manipulate the load allocation algorithm in their own benefit and their selfish behavior may lead to severe performance degradation and poor efficiency. Solving such problems involving selfish agents is the object of *mechanism design theory* (also called *implementation theory*)[10]. This theory helps design protocols in which the agents are always forced to tell the truth and follow the rules. Such mechanisms are called *truthful* or *strategy-proof*.

Each participating agent has a privately known function called *valuation* which quantifies the agent benefit or loss. The valuation depends on the outcome and is reported to a centralized mechanism. The mechanism chooses an outcome that maximizes a given objective function and makes payments to the agents. The valuations and payments are expressed in some common unit of currency. The payments are designed and used to motivate the agents to report their true valuations. Reporting the true valuations leads to an optimal value for the objective function. The goal of each agent is to maximize the sum of her valuation and payment.

In this paper we consider the mechanism design problem for load balancing in distributed systems. A general formulation of the *load balancing problem* is as follows: given a large number of jobs, find the allocation of jobs to computers optimizing a given objective function (e.g. total execution time). Our goal is to design a mechanism that uses the optimal load balancing algorithm. The optimal algorithm belongs to the global approach in the classification presented in [4]. To design our mechanism we use the framework derived by Archer and Tardos in [1]. We assume that each computer in the distributed system is characterized by its processing rate and only computer i knows the true value of its processing rate. Jobs arrive at the system with a given arrival rate. The optimal algorithm finds the fraction of load that is allocated to each computer such that the expected execution time is minimized. The *cost* incurred by each computer is proportional to its utilization. The mechanism will ask each agent (computer) to report its processing rate and then compute the allocation using the optimal algorithm. After computing the allocation the mechanism hands payments to computers. Each computer goal is to chose a processing rate to report to the mechanism such that its profit is maximized. The *profit* is the difference between the payment handed by the mechanism and the true cost of processing the allocated jobs. The payments handed by the mechanism must motivate the computers to report their true value such that the expected response time of the entire system is minimized. Thus we need to design a payment function that guarantees this property.

Related work

Recently, with the emergence of the Internet as a global platform for computation and communication, the need for efficient protocols that deals with self-interested agents has increased. This motivated the use of mechanism design theory in different settings such as market based protocols for resource allocation in computational grids [3, 13], market based protocols for scheduling [12], congestion control [5] and mechanisms for trading CPU time [8]. For an introduction to general mechanism design theory see [10]. The most important result in mechanism design theory is the Vickrey-Clarke-Groves (VCG) mechanism [10]. The VCG mechanism allows arbitrary form for valuations and its applicability is restricted to utilitarian objective functions (i.e. the objective function is the sum of agents' valuations). Nisan and Ronen [9] studied the mechanism design problem for several standard problems in computer science. Using the VCG mechanism they solved the shortest path problem in a graph where each edge belongs to a different agent. For scheduling on unrelated machines they designed an n-approximation truthful mechanism, where n is the number of agents. Archer and Tardos [1] applied mechanism design theory to several combinatorial optimization problems where agent's secret data is represented by a single real valued parameter. They provided a method to design mechanisms for general objective functions and restricted form for valuations. For scheduling related parallel machines they gave a 3-approximation algorithm and used it to design a truthful mechanism. They also gave truthful mechanisms for maximum flow, scheduling related machines to minimize the sum of completion times, optimizing an affine function and special cases of uncapacitated facility location.

Our results

We design a truthful mechanism for solving the static load balancing problem in distributed systems. We prove that using the optimal allocation algorithm the output function admits a truthful payment scheme satisfying voluntary participation. We derive a protocol that implements our mechanism and present experiments to show its effectiveness.

Organization

The paper is structured as follows. In Section 2 we present the mechanism design terminology. In Section 3 we present our distributed system model. In Section 4 we design a truthful mechanism for load balancing in distributed systems. In Section 5 the effectiveness of our load balancing mechanism is investigated. In Section 6 we draw conclusions and present future directions.

2. Mechanism Design Concepts

In this section we introduce some important mechanism design concepts. We limit our description to mechanism design problems for one parameter agents. In this type of mechanism design problems each agent has some private data represented by a single real valued parameter [1]. In the following we define such problem.

Definition 2.1 (Mechanism design problem) A *mechanism design problem for one parameter agents* is characterized by:

(i) A finite set Λ of allowed outputs. The output is a vector $\lambda(b) = (\lambda_1(b), \lambda_2(b), \ldots, \lambda_n(b))$, $\lambda(b) \in \Lambda$, computed according to the agents' bids, $b = (b_1, b_2, \ldots, b_n)$. Here, b_i is the value (bid) reported by agent i to the mechanism.

(ii) Each agent i, $(i = 1, \ldots, n)$, has a privately known parameter t_i called her *true value*. The cost incurred by each agent depends on the output and on her true value and is denoted as $cost_i$.

(iii) Each agent goal is to maximize her profit. The profit of agent i is $profit_i = P_i(b) - cost_i(b)$, where P_i is the payment handed by the mechanism to agent i.

(iv) The goal of the mechanism is to select an output λ that optimizes a given cost function $g(b, \lambda)$. $\square$

We assume that the cost functions have the following particular form: $cost_i(t_i, \lambda) = t_i \lambda_i$, i.e. t_i represents the cost per unit load.

Definition 2.2 (Mechanism) A *mechanism* is characterized by two functions:

(i) The *output function* $\lambda(b) = (\lambda_1(b), \lambda_2(b), \ldots, \lambda_n(b))$. This function has as input the vector of agents' bids $b = (b_1, b_2, \ldots, b_n)$ and returns an output $\lambda \in \Lambda$.

(ii) The *payment function* $P(b, \lambda)$ that gives the payment handed by the mechanism to each agent. $\square$

Notation: In the rest of the paper we denote by b_{-i} the vector of bids not including the bid of agent i. The vector b is represented as (b_{-i}, b_i).

Definition 2.3 (Truthful mechanism) A mechanism is called *truthful* if for every agent i of type t_i and for every bids b_{-i} of the other agents, the agent's profit is maximized when she declares her real type t_i. (i.e. truth-telling is a dominant strategy). $\square$

Definition 2.4 (Truthful payment scheme) We say that an output function admits a *truthful payment scheme* if there exists a payment function P such that the mechanism is truthful. $\square$

A desirable property of a mechanism is that the profit of a truthful agent is always non-negative. The agents hope for a profit by participating in the mechanism.

Definition 2.5 (Voluntary participation mechanism) We say that a mechanism satisfies the *voluntary participation condition* if $profit_i(t_i, (b_{-i}, t_i)) \geq 0$ for every agent i, true values t_i, and other agents' bids b_{-i} (i.e. truthful agents never incurs a loss). $\square$

3. Distributed System Model

We consider a distributed system that consists of n heterogeneous computers connected by a communication network. Computers have the same processing capabilities in the sense that a job may be processed from start to finish at any computer in the system. We assume that each computer is modeled as an M/M/1 queueing system (i.e. Poisson arrivals and exponentially distributed processing times) [6] and is characterized by its *average processing rate* μ_i, $i = 1, \ldots, n$. Jobs are generated by users and arrive at the system according to a time invariant Poisson process with average rate Φ. We call Φ the *total job arrival rate* in the system. The total job arrival rate must be less than the aggregate processing rate of the system (i.e. $\Phi < \sum_{i=1}^{n} \mu_i$). The system has to decide on how to distribute jobs to computers such that it will operate optimally. We assume that the decision to distribute jobs to computers is *static* i.e. it does not depend on the current state of the system. Thus we need to find the *load* λ_i that is assigned to computer i $(i = 1, \ldots, n)$ such that the expected response time of all jobs is minimized. The expected response time at computer i is given by:

$$F_i(\lambda_i) = \frac{1}{\mu_i - \lambda_i} \tag{1}$$

Thus the overall expected response time is given by:

$$D(\lambda) = \frac{1}{\Phi} \sum_{i=1}^{n} \lambda_i F_i(\lambda_i) = \frac{1}{\Phi} \sum_{i=1}^{n} \frac{\lambda_i}{\mu_i - \lambda_i} \tag{2}$$

where $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ is the vector of loads assigned to computers.

We assume that computers are agents and each of them has a *true value* t_i represented by the inverse of its processing rate, $t_i = \frac{1}{\mu_i}$. Only computer i knows t_i. The mechanism will ask each computer i to report its value b_i (the inverse of its processing rate). The computers may not report the true value. After all the computers report their values the mechanism computes an output function $\lambda(b) = (\lambda_1(b), \lambda_2(b), \ldots, \lambda_n(b))$ according to the agents' bids such that the overall expected execution time is minimized. The mechanism also hands a payment $P_i(b)$ to each computer. All computers know the algorithm used to compute the output function (allocation) and the payment scheme.

Each computer incurs some cost: $cost_i = t_i \lambda_i(b)$. The cost is equivalent to computer utilization. The greater the utilization, the greater the cost. We assume each computer wants to choose its strategy (what value b_i to report) such that its profit is maximized. The profit for each computer is defined as the payment received from the mechanism minus the cost incurred in running the jobs allocated to it: $profit_i = P_i(b) - cost_i(b)$.

Our goal is to design a truthful mechanism that minimizes the overall expected response time of the system. This involves finding an allocation algorithm and a payment scheme that minimizes the overall expected response time according to the computer bids b_i and motivates all the computers to report their true values t_i.

4. Designing the Mechanism

To design our load balancing mechanism we use the framework proposed by Archer and Tardos in [1]. They provided a method to design mechanisms where each agent's true data is represented by a single real valued parameter. According to this method, to obtain a truthful mechanism we must find an output function satisfying two conditions: (a) it minimizes $D(\lambda)$ and, (b) it is decreasing in the bids. In addition, we want a mechanism satisfying voluntary participation. To guarantee this property we must find a payment function satisfying voluntary participation.

First we are interested in finding an output function $\lambda(b)$ that minimizes the expected execution time over all jobs, $D(\lambda)$, and produces a feasible allocation. Then we will show that this output function is decreasing in the bids.

Definition 4.1 (Feasible allocation) A *feasible allocation* $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ is a load allocation that satisfies the following conditions:
  (i) Positivity: $\lambda_i \geq 0$, $i = 1, \ldots, n$;
  (ii) Conservation: $\sum_{i=1}^{n} \lambda_i = \Phi$;
  (iii) Stability: $\lambda_i < \mu_i$, $i = 1, \ldots, n$. $\square$

The optimal load allocation can be obtained solving the following nonlinear optimization problem: $\min_\lambda D(\lambda)$ subject to the constraints defined by the conditions i)-iii).

Thus, obtaining the solution to this problem involves minimizing the convex function $D(\lambda)$ over a convex feasible region defined by the conditions i)-iii). In this case the first order Kuhn-Tucker conditions are necessary and sufficient for optimality [7].

Let $\alpha \geq 0$, $\eta_i \geq 0$, $i = 1, \ldots, n$ denote the Lagrange multipliers [7]. The Lagrangian function is: $L(\lambda_1, \lambda_2, \ldots, \lambda_n, \alpha, \eta_1, \ldots, \eta_n) = \sum_{i=1}^{n} \lambda_i F_i(\lambda_1) - \alpha(\sum_{i=1}^{n} \lambda_i - \Phi) - \sum_{i=1}^{n} \eta_i \lambda_i$.

The Kuhn-Tucker conditions imply that λ_i, $i = 1, \ldots, n$ is the optimal solution to our problem if and only if there exists $\alpha \geq 0$, $\eta_i \geq 0$, $i = 1, \ldots, n$ such that:

$$\frac{\partial L}{\partial \lambda_i} = 0 \qquad \frac{\partial L}{\partial \alpha} = 0 \tag{3}$$

$$\eta_i \lambda_i = 0, \quad \eta_i \geq 0, \quad \lambda_i \geq 0, \qquad i = 1, \ldots, n \tag{4}$$

These conditions become:

$$\alpha = \lambda_i F_i'(\lambda_i) + F_i(\lambda_i), \quad \text{if} \quad \lambda_i > 0 \quad 1 \leq i \leq n \tag{5}$$

$$\alpha \leq \lambda_i F_i'(\lambda_i) + F_i(\lambda_i), \quad \text{if} \quad \lambda_i = 0 \quad 1 \leq i \leq n \quad (6)$$

$$\sum_{i=1}^{n} \lambda_i = \Phi, \quad \lambda_i \geq 0, \quad i = 1, \ldots, n \quad (7)$$

By solving these conditions one can obtain the optimal algorithm but this is not our focus in this paper. Algorithms for this kind of optimization problem were proposed in the past [11]. For the clarity of presentation we describe a variant of the algorithm in [11] using our notations. Our derivation of this algorithm is different from their approach and was partially presented above because some of the equations will be used to prove our results in Theorem 4.1 and Theorem 4.2 below. We now present the algorithm.

OPTIM algorithm:

> **Input:** Bids submitted by computers: $b_1, b_2, \ldots b_n$;
> Total arrival rate: Φ
> **Output:** Load allocation: $\lambda_1, \lambda_2, \ldots \lambda_n$;
> 1. Sort the computers in increasing order of their bids ($b_1 \leq b_2 \leq \ldots \leq b_n$);
> 2. $c \leftarrow \dfrac{\sum_{i=1}^{n} 1/b_i - \Phi}{\sum_{i=1}^{n} \sqrt{1/b_i}}$;
> 3. **while** ($c > \sqrt{1/b_n}$) **do**
> $\lambda_n \leftarrow 0; n \leftarrow n - 1;$
> $c \leftarrow \dfrac{\sum_{i=1}^{n} 1/b_i - \Phi}{\sum_{i=1}^{n} \sqrt{1/b_i}}$;
> 4. **for** $i = 1, \ldots, n$ **do**
> $\lambda_i \leftarrow 1/b_i - c\sqrt{1/b_i}$;

This algorithm computes an allocation $\lambda(b) = (\lambda_1(b), \lambda_2(b), \ldots, \lambda_n(b))$ that provides the overall optimum for the expected execution time. This optimum is obtained according to the bids reported by computers. If some of the computers declare values different than their true values ($t_i = 1/\mu_i$), this optimum may not be the same as the 'true optimum' obtained when all the computers declare their true values. So if some of the computers lie we expect worse performance (i.e. higher overall expected execution time).

We found an output function (given by OPTIM) that minimizes $D(\lambda)$. Now we must prove that this output function admits a truthful mechanism. In the following we state two theorems: (i) that the output function is decreasing in the bids (thus guaranteeing truthfulness) and, (ii) that our mechanism admits a truthful payment scheme satisfying voluntary participation. Due to space limitation the proofs of these two theorems are not presented here. They will be presented in the full version of the paper.

Definition 4.2 (Decreasing output function) An output function $\lambda(b) = (\lambda_1(b), \lambda_2(b), \ldots, \lambda_n(b))$ is *decreasing* if each $\lambda_i(b_{-i}, b_i)$ is a decreasing function of b_i for all i and b_{-i}. $\square$

Theorem 4.1 The output function $\lambda(b)$ computed by the optimal algorithm is decreasing. $\square$

Theorem 4.2 The output function $\lambda(b)$ computed by the optimal algorithm admits a *truthful* payment scheme satisfying *voluntary participation* and the payment for each computer i ($i = 1, 2, \ldots, n$) is given by:

$$P_i(b_{-i}, b_i) = b_i \lambda_i(b_{-i}, b_i) + \int_{b_i}^{\infty} \lambda_i(b_{-i}, x) dx \quad (8)$$

$\square$

Remark: Analogously to [1] we obtained $P_i(b_{-i}, b_i)$ for our mechanism. The first term, $b_i \lambda_i(b_{-i}, b_i)$, of the payment function in equation (8) compensates the cost incurred by computer i. The second term, $\int_{b_i}^{\infty} \lambda_i(b_{-i}, x) dx$ represents the expected profit of computer i. If computer i bids its true value, t_i, then its profit is: $profit_i(b_{-i}, t_i) = t_i \lambda_i(b_{-i}, t_i) + \int_{t_i}^{\infty} \lambda_i(b_{-i}, x) dx - t_i \lambda_i(b_{-i}, t_i) = \int_{t_i}^{\infty} \lambda_i(b_{-i}, x) dx$.

Because the optimal algorithm assumes a central dispatcher, the mechanism will be implemented in a centralized way as part of the dispatcher code. We assume that the dispatcher is run on one of the computers and is able to communicate with all the other computers in the distributed system.

In the following we present the protocol that implements our load balancing mechanism (LBM). This protocol has two phases: bidding and completion.

Protocol LBM:

> Phase I: Bidding
> 1. The dispatcher sends a request for bids message (*ReqBid*) to each computer in the system.
> 2. When a computer receives a *ReqBid* it replies with its bid b_i to the dispatcher.
>
> Phase II: Completion
> 1. After the dispatcher collects all the bids it does the following:
>   1.1. Computes the allocation using OPTIM algorithm.
>   1.2. Computes the payments P_i for each computer using equation (8).
>   1.3. Sends P_i to each computer i.
> 2. Each computer receives its payment and evaluates its profit.

This protocol is executed periodically or when there is a change in the total job arrival rate. During two executions of this protocol the jobs will be allocated to computers by the dispatcher according to the allocation computed by OPTIM. Computers will receive the maximum profit only when they report the true value.

5. Experimental results

To study the effectiveness of our truthful mechanism we simulated a heterogeneous system consisting of 16 computers with four different processing rates. In Table 1 we

present the system configuration. The first row contains the relative processing rates of each of the four computer types. Here, the relative processing rate for computer C_i is defined as the ratio of the processing rate of C_i to the processing rate of the slowest computer in the system. The second row contains the number of computers in the system corresponding to each computer type. The last row shows the processing rate of each computer type in the system.

Relative processing rate	1	2	5	10
Number of computers	6	5	3	2
Processing rate (jobs/sec)	0.013	0.026	0.065	0.13

Table 1. System configuration.
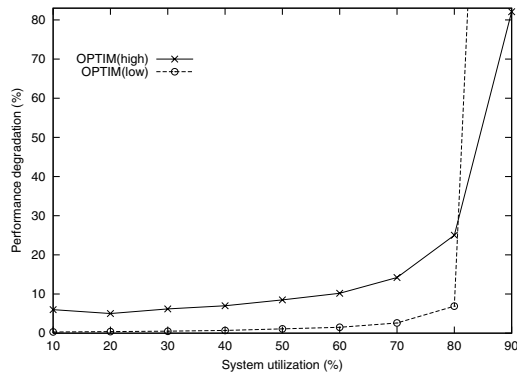
Figure 1. Performance degradation vs. system utilization.

Figure 2. Profit for each computer (medium system load)
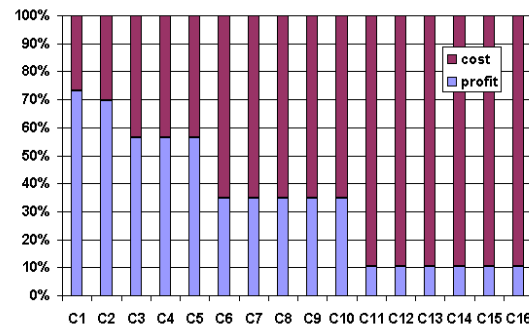
Figure 3. Payment structure for each computer (C_1 bids higher)

We study the performance degradation due to false bids declarations. We define *performance degradation(PD)* as follows: $PD = ((D_F - D_T)/D_T)100\%$, where D_F is the response time of the system when one or more computers report false values; and D_T is the response time of the system when all computers report their true value. PD quantifies the increase in the execution time due to false bidding. We expect an increase in the response time and PD when one or more computers lie.

In our experiments we consider that the fastest computer C_1 declares false bids. C_1 has $t_1 = 1/\mu_1 = 7.69$ as its true value. In Figure 1 we present the degradation in expected response time of the system for different values of system utilization (ranging from 10% to 90%) and two types of bidding: overbidding and underbidding. *System utilization* (ρ) is defined as the ratio of total arrival rate to aggregate processing rate of the system: $\rho = \frac{\Phi}{\sum_{i=1}^{n} \mu_i}$. In the first experiment, C_1 bids 7% lower than the true value. In this case the performance degradation is around 2% for low and medium system utilization, increasing drastically (around
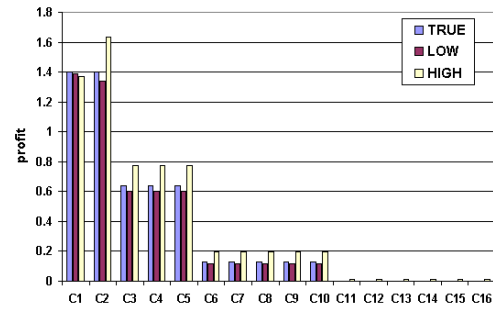
300%) for high system utilization. This increase is due to computer C_1 overloading. The overloading occurs because C_1 bids lower, that means it reports a higher value for its processing rate. The algorithm will allocate more jobs to C_1 increasing its response time. This increase is reflected in the expected response time of the system. In the second experiment, C_1 bids 33% higher than the true value. In this case the performance degradation is about 6% at low system utilization, about 15% at medium system utilization and more than 80% at high system utilization. It can be observed that small deviations from the true value of only one computer may lead to large values of performance degradation. If we consider that more than one computer does not report its true value then we expect very poor performance. This justifies the need for a mechanism that will force the computers to declare their true values.

In Figure 2 we present the profit for each computer at medium system loads ($\rho = 50\%$). It can be observed that the profit at C_1 is maximum if it bids the true value, 3% lower if it bids higher and 1% lower if it bids lower. The
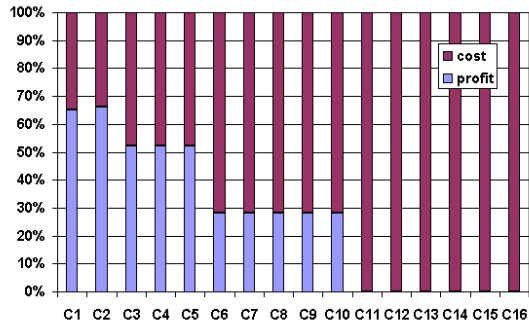
Figure 4. Payment structure for each computer (C_1 bids lower)

mechanism penalizes C_1 if it does not report the true value. When C_1 bids lower the other computer's profits are lower because their payments decrease. Computers C_{11} to C_{16} are not utilized when C_1 underbids and when it reports the true value, thus they will not gain anything ($profit_i = 0$, $i = 11, 12, \ldots, 16$). These computers will be utilized in the case when C_1 overbids, getting a small profit. When C_1 overbids the profit for all the computers except C_1 is higher than in the case when C_1 bids the true value. This is because the payments increase for these computers.

An important issue is the *frugality* of our mechanism. We say that a mechanism is *frugal* if the mechanism's payments are small by some measure [2]. This property gives us an idea of how efficient a mechanism is. The mechanism is interested in keeping its payments as small as possible. Our mechanism must preserve voluntary participation, so the lower bound on its payments is the total cost incurred by the computers. In Figure 3 and 4 we present the cost and profit as fractions of the payment received by each computer at medium loads. It can be observed that the cost incurred by C_1 when it bids higher is about 25% of the payment. In the case when C_1 bids lower its cost is about 35% of the payment. For the other computers the cost is between 50% and 90% when C_1 bids higher and between 53% and 100% when C_1 bids lower. For the distributed system considered in these experiments (medium loads) the highest payment given to a computer is about 3 times its cost.

6. Conclusion

In this paper we investigated the problem of designing protocols for resource allocation involving selfish agents. Solving this kind of problems is the object of mechanism design theory. Using this theory we designed a truthful mechanism for solving the load balancing problem in heterogeneous distributed systems. We proved that using the optimal allocation algorithm the output function admits a truthful payment scheme satisfying voluntary participation. We derived a protocol that implements our mechanism and presented experiments to show its effectiveness. Future work will address the implementation of our load balancing mechanism in a real distributed system as well as the design of a distributed load balancing mechanism.

References

[1] A. Archer and E. Tardos. Truthful mechanism for one-parameter agents. In *Proc. of the 42nd IEEE Symp. on Foundations of Computer Science*, pages 482–491, October 2001.

[2] A. Archer and E. Tardos. Frugal path mechanisms. In *Proc. of the 13th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 991–999, January 2002.

[3] R. Buyya, D. Abramson, and J. Giddy. A case for economy grid architecture for service-oriented grid computing. In *Proc. of the 10th IEEE Heterogeneous Computing Workshop*, April 2001.

[4] D. Grosu, A. T. Chronopoulos, and M. Y. Leung. Load balancing in distributed systems: An approach using cooperative games. In *Proc. of the 16th International Parallel and Distributed Processing Symposium*, April 2002.

[5] R. Karp, E. Koutsoupias, C. H. Papadimitriou, and S. Shenker. Optimization problems in congestion control. In *Proc. of the 41st IEEE Symp. on Foundations of Computer Science*, pages 66–74, November 2000.

[6] L. Kleinrock. *Queueing Systems - Volume 1: Theory*. John Wiley and Sons, 1975.

[7] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, Reading, Mass., 1984.

[8] N. Nisan, S. London, O. Regev, and N. Camiel. Globally distributed computation over Internet - The POPCORN project. In *Proc. of the 18th IEEE International Conference on Distributed Computing Systems*, pages 592–601, May 1998.

[9] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proc. of the 31rd Annual ACM Symp. on Theory of Computing (STOC' 1999)*, pages 129–140, May 1999.

[10] M. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, Cambridge, Mass., 1994.

[11] X. Tang and S. T. Chanson. Optimizing static job scheduling in a network of heterogeneous computers. In *Proc. of the Intl. Conf. on Parallel Processing*, pages 373–382, August 2000.

[12] W. E. Walsh, M. P. Wellman, P. R. Wurman, and J. K. MacKie-Mason. Some economics of market-based distributed scheduling. In *Proc. of the 18th IEEE International Conference on Distributed Computing Systems*, pages 612–621, May 1998.

[13] R. Wolski, J. S. Plank, T. Bryan, and J. Brevik. G-commerce: market formulations controlling resource allocation on the computational grid. In *Proc. of the 15th IEEE International Parallel and Distributed Processing Symposium*, April 2001.