5-2003

Computational Markets to Regulate Mobile-Agent Systems

Jonathan Bredin
Dartmouth College

David Kotz
Dartmouth College

Daniela Rus
Dartmouth College

Rajiv T. Maheswaran
University of Illinois

Cagri Imer
University of Illinois

See next page for additional authors


Autonomous Agents and Multi-Agent Systems, 6, 235–263, 2003

Computational Markets to Regulate Mobile-Agent Systems

JONATHAN BREDIN, DAVID KOTZ, AND DANIELA RUS jbredin@coloradocollege.edu
Department of Computer Science, Dartmouth College, Hanover, NH 03755

RAJIV T. MAHESWARAN, CAGRI IMER, AND TAMER BASAR
Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801

Abstract. Mobile-agent systems allow applications to distribute their resource consumption across the network. By prioritizing applications and publishing the cost of actions, it is possible for applications to achieve faster performance than in an environment where resources are evenly shared. We enforce the costs of actions through markets, where user applications bid for computation from host machines.

We represent applications as collections of mobile agents and introduce a distributed mechanism for allocating general computational priority to mobile agents. We derive a bidding strategy for an agent that plans expenditures given a budget, and a series of tasks to complete. We also show that a unique Nash equilibrium exists between the agents under our allocation policy. We present simulation results to show that the use of our resource-allocation mechanism and expenditure-planning algorithm results in shorter mean job completion times compared to traditional mobile-agent resource allocation. We also observe that our resource-allocation policy adapts favorably to allocate overloaded resources to higher priority agents, and that agents are able to effectively plan expenditures, even when faced with network delay and job-size estimation error.

Keywords: mobile agents, market-based control, resource allocation

1. Introduction

We develop self-regulating frameworks for networked applications where users may have conflicting interests and differing priorities. In particular, we are interested in market-based structures where software agents compete for computational resources. In our system, local auctions prioritize agents in a distributed environment where communication costs may be high. We observe that the cost of our prioritization is small, and that the algorithms that drive an agent's bidding strategy are robust to errors the agent's job requirements.

1.1. Computational-resource markets

We use a currency-based resource-allocation policy, where agents buy access to computational resources. The policy has a straightforward ideal: the cost of resource access is proportional to the quantity allocated. Our allocation policy partitions each resource independently, to satisfy the demand of all interested agents, such that no agent could benefit from altering its bid. We propose a mechanism to collect agents'

bids as functions, and derive a bidding strategy that minimizes the computational latency in an agent's execution.

More specifically, we present a model that captures the priority of the various tasks in a network. The model uses an allocation policy that respects priorities, and requires that each agent need only know its own preferences regarding task completion.

Ideas from economics provide solutions to many of the issues related to coordinating access to resources in distributed applications. Market-based resource allocation has been used in various distributed resource-allocation problems including computing resources, network bandwidth, and manufacturing systems [9]. For example, auction mechanisms have been used to allocate computer resources [11], and microeconomic approaches can distributedly allocate resources [16]. There are many models where servers sell resources to agents. The price of a resource reflects congestion, and serves as a load-balancing mechanism [7, 27].

Computational markets rely on some form of currency exchange. Currency represents an agent's potential to act in the network. It is possible for agents to exchange electronic legal tender implemented through cryptographic verification [12, 22]. In this paper, however, we consider operation within a closed environment where the system administrator uses currency to compute a fair allocation among the users.

1.2. Motivational applications

To study how effective are computational markets techniques for self-regulation, we focus on applications that require careful latency management. This is a timely area of investigation, as the proliferation of small-scale computing devices has lead to large-scale wired and wireless networks, yet progress to reduce network latency has not kept pace with the developments of high-performance computing, and the improvement of network bandwidth. As computers have become more economical and standardized, computation is no longer the constraining factor in distributed-application performance. Increasingly, communication costs of data access comprise the bulk of an application's execution time. The disparity between the advances in computation and communication latency is exaggerated by the greater use of wireless and intermittently connected networks. A technique that reduces an application's end-to-end latency, is to move computation closer to scarcer resources. This reorganization can avoid latency incurred in network communication.

A mobile agent is a user program with the ability to autonomously move from one host to resume execution at another. A mobile-agent system provides the mechanisms for decentralizing resource allocation by relocating computation represented by mobile agents. By moving the computation (a mobile agent) to the data, bandwidth usage and completion time are optimized. Rapidly evolving networks, where nodes and features may be added or removed often, are easily implemented using mobile-agent systems. A challenge involved in implementing applications within a mobile-agent system is implementing a structure that incorporates the idea that different tasks in a system possess different priorities. The amount and quality of the resources allocated to an agent should reflect the agent's priority. Computational

markets, and especially auctions, are thus naturally suited for controlling resource access in applications implemented with mobile-agent systems.

1.3. Mobility benefits

Mobile-agent systems provide a decentralized flexible architecture on which to base network management applications [3] and handle user preferences [15]. Mobile agents have been used to improve fault tolerance by applying knowledge of the loads on various resources, and resubmitting tasks to alternate sites [19, 24]. Other applications for mobile agents are to enhance video-conferencing performance [2]; to allow users to more efficiently operate on remote distributed data on an unreliable network [14]; and as an alternative to client-server networking [20].

Mobile agents eliminate much of the need for static protocols and allow networks to grow and change seamlessly. The acceptance of Java on many platforms, and the current efforts to standardize mobile agents [21] lead us to believe that a standard for mobile agents will emerge, facilitating their use on almost any type of network.

1.4. Allocation scenario

In this paper we address an economic system where the players are mobile agents and resource contention drives a hierarchy of distributed resource-allocation decisions. The mobile agents implement distributed applications. We restrict our attention to systems such as military and corporate intranets where users have already passed a higher-level admission procedure. Each user has task sequences to execute. To carry out a task sequence, the user creates a mobile agent and allocates some endowment to it, which the agent cannot return to the user. The agent's job is to select the set of resources required for execution of its tasks, and to budget its endowment to negotiate priority for each resource. Its goal is to minimize the time to complete all of its tasks.

For example, consider the scenario where the network is a subnet of computers at a research lab, and resources are CPUs at nodes throughout the network. Figure 1 illustrates an example task sequence for a mobile agent. The agent can relocate to one of several hosts to retrieve an image from a database or device, then jump to another host to process the image, and finally move to another host to render and display the results.

The agent must decide which hosts to visit, as well as execution priority at each host. The focus of this paper, is how an agent can budget its endowment over a sequence of tasks. For evaluation of our allocation policy, we provide methods to estimate cost and performance, and an algorithm to choose an agent's path. Budget construction requires that each agent negotiates with other agents at the site for access. To this end, we establish a mechanism that allows each agent to prioritize its execution to minimize its own completion time, that respects the relative priority of other agents. Each resource is indifferent to how it is partitioned among the

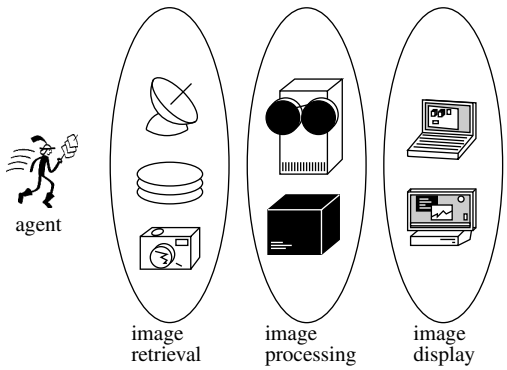

Figure 1. An example mobile-agent itinerary. The agent must visit one host from each group and choose at what priority to execute at each visited host.

agents, as long as it is fully utilized. An agent receives exclusive access for an idle resource. When multiple agents simultaneously access a resource, agents participate in a bidding process to divide the resource among themselves.

1.5. Paper organization

The bidding and allocation mechanisms are described in Section 2. In Section 3 we derive a response to our mechanism that optimizes an agent's execution time in the absence of market response to the agent's demand. We extend the strategy in Section 4 to show how to compute an allocation that satisfies all agents' response functions for a single resource. The resulting allocation is a Nash equilibrium among the agents that request access. In Section 5, we prove the uniqueness of the equilibrium. The price that results from the allocation, serves as an indicator for congestion or demand that prospective consumer agents use to budget their expenditures.

In Section 6, we discuss how our allocation mechanism would be implemented in a network. In Section 7, we show simulation results that demonstrate that systems that use our allocation policy exhibit higher throughput than ones that use a traditional mobile-agent computation-allocation policy. We also compare our policy with one that optimizes throughput to show that the cost of prioritizing agents is small. Finally, we observe that our allocation and planning algorithms are insensitive to agents' errors in job-size estimation, and are no more sensitive to network delay than are traditional policies.

We describe some related work in Section 8. In Section 9, we discuss our results and identify directions for future research. We provide two appendices. One proves the concavity of the agents' bid function, and the other is a table of notations used in the paper.

2. System model

We study an environment where prioritized agents each have a sequence of computational tasks. In this section, we describe task requirements and how a user prioritizes her agents. We also define the interface that agents use to negotiate access priority when there is contention among multiple agents.

2.1. Itinerary description

A user creates an agent to complete a task sequence as quickly as possible. Each task requires access to only one resource, which is possibly replicated at several sites in the network. We denote the size of the kth task in the ith agent's itinerary as q_k^i. For the image retrieval example in Section 1, task size is the expected number of CPU clock cycles required to complete the task.

An agent chooses a resource immediately prior to execution of the corresponding task. Resource capacity and the agent's resource-congestion estimates for all prospective resources influence selection. In our analysis, we assume that network delay between resource locations is fixed.

Once the agent selects a resource, the agent commits to use the resource to finish the current task. Let c_k^i denote the fixed capacity of the resource used by the ith agent's kth task. With respect to our subnet example, c_k^i is the clock speed of the CPU at the computer where the ith agent's kth computation is completed.

2.2. Priority

The user expresses the priority of an agent's itinerary by endowing it with electronic currency to compete for network resources. The ith agent's endowment is e^i dollars, and cannot be returned to the user. We assume that the user determines e^i through a higher-level optimization problem based on knowledge of previous job completion times given various endowments. We take e^i as a given in this paper. If the ith agent has K^i tasks and $q^i = \sum_{k=1}^{K^i} q_k^i$, then $\rho^i = e^i/q^i$, the dollars endowed per-unit job, is a measure of the agent's priority, or more accurately, the priority of the task sequence to its user.

A larger ρ^i represents an ability to purchase larger portions of resources for each job-unit to be completed, and enables the agent to finish more quickly. It could indicate that the particular task sequence is more important (emergency messaging versus routine maintenance), or that the user is of higher priority to the network, and has more capital to spend.

2.3. Allocation policy

An agent receives exclusive access, without effect to its endowment, to an uncontended resource. When multiple agents request a resource, however, the resource

decides how to partition its capacity. We assume that access to a resource can be arbitrarily divided, without switching overhead, among many agents. The resource recalculates the allocation whenever an agent arrives at or departs.

The mechanism that we present follows the paradigm that all agents that request access to a resource are charged the same rate per-unit of capacity allocated. Using our formalism, if the ith agent pays u_k^i dollars per second, and θ_k denotes the total amount that all agents currently pay, then the ith agent will receive service at a rate v_k^i given by:

$$v_k^i = c_k^i \left(\frac{u_k^i}{u_k^i + \theta_k^{-i}} \right), \tag{1}$$

where $\theta_k^{-i} := \theta_k - u_k^i$ represents the sum of payment rates from all agents except the ith agent. For example, if a resource accommodates three agents that pay one, two, and three dollars per second, respectively, the agents will receive access to one sixth, one third, and one half of the resource capacity.

If the ith agent receives service at rate v_k^i, the time required to execute the kth task is:

$$t_k^i = \frac{q_k^i}{v_k^i} = \frac{q_k^i (u_k^i + \theta_k^{-i})}{c_k^i u_k^i}, \tag{2}$$

and the cost of completing the kth task is:

$$m_k^i = u_k^i t_k^i = \frac{q_k^i (u_k^i + \theta_k^{-i})}{c_k^i}. \tag{3}$$

Ultimately, the ith agent expresses its request to the kth resource through a bid that is a function that returns u_k^i, the amount an agent will pay, conditioned on θ_k, the aggregate amount that all agents pay for access. The class of functions that ensure that an allocation is feasible and unique is described in Section 5.

The resource owner calculates a price, represented by θ_k, that satisfies all agents' bid functions every time an agent expresses new interest in, or relinquishes access to, the resource. Each interested agent submits a new bid with updated cost, and performance estimates of resources the agent plans to use in the future. The resource ignores agents that pay nothing in equilibrium until another chance to reallocate arises.

In Section 3, we calculate the optimal bidding function for one agent under the condition that all other agents' payments are fixed throughout the network. In Section 4, we show that there exists a Nash equilibrium among agents using bidding functions in the form of the optimal response, and in Section 5 we show that the equilibrium is unique. Thus, we have an allocation policy that uniquely satisfies local agents' response functions.

3. Single-agent optimization

To determine how an agent should bid, we simplify the problem by fixing the payments of all the other agents in the network, and we assume that the ith agent has perfect information regarding θ_k^{-i} for all hosts that it will visit. The agent's objective is to minimize its total execution time, and it is constrained by its endowment. Since t_k^i is a strictly decreasing function of u_k^i, we expect that the agent's expenditure at the solution should be at the boundary of its constraint region, i.e., it will minimize its completion time if it spends the entire endowment.

We formulate the ith agent's problem as:

$$\min \sum_{k=1}^{K^i} t_k^i \qquad \text{s.t.} \qquad \sum_{k=1}^{K^i} m_k^i \leq e^i, \tag{4}$$

which we solve with Lagrangian methods by defining the Lagrangian as:

$$\mathcal{L} = \sum_{k=1}^{K^i} t_k^i + \lambda\left(\sum_{k=1}^{K} m_k^i - e^i\right). \tag{5}$$

Substituting for t_k^i and m_k^i into Equation 5, and taking partial derivatives with respect to u_k^i, we arrive at:

$$\frac{\partial \mathcal{L}}{\partial u_k^i} = \frac{-q_k^i \theta_k^{-i}}{c_k^i u_k^{i^2}} + \lambda \frac{q_k^i}{c_k^i} = 0 \quad \Rightarrow \quad \lambda = \frac{\theta_k^{-i}}{u_k^{i^2}}. \tag{6}$$

Since we are dealing with the ith agent's decision, to simplify notation, we drop i superscripts, except in θ_k^{-i}, which we use to denote the sum of competing agents' bids at the kth server agent i visits.

Note that $\theta_k^{-i} > 0$ implies $\lambda > 0$ for all but the trivial case when only one agent bids. Thus, we have the following relationship between any two bids, j and k:

$$u_k = u_j \sqrt{\frac{\theta_k^{-i}}{\theta_j^{-i}}}. \tag{7}$$

The relationship states that as contention for one resource increases, the agent spends more for that resource, but the increase is sublinear with respect to the contention. Combining the inequality constraint and Equation 7, we get

$$\lambda \frac{\partial \mathcal{L}}{\partial \lambda} = \lambda\left(\sum_{k=1}^{K} \frac{q_k(u_k + \theta_k^{-i})}{c_k} - e\right) = 0. \tag{8}$$

Since $\lambda > 0$, it follows that the inequality constraint must be satisfied with equality, which shows that the total expenditure is on the boundary of the constraint

as we expected. Substituting for $\{u_k\}_{k=2}^{K}$ in terms of u_1 using the relationship in Equation 7, we have:

$$\frac{q_1}{c_1}(u_1 + \theta_1^{-i}) + \sum_{k \neq 1} \frac{q_k}{c_k} \sqrt{\frac{\theta_k^{-i}}{\theta_1^{-i}}} u_1 + \sum_{k \neq 1} \frac{q_k}{c_k} \theta_k^{-i} - e = 0. \tag{9}$$

Solving the previous equation for u_1, we get

$$u_1 = \frac{e - \sum_{k \neq 1} \frac{q_k}{c_k} \theta_k^{-i} - \frac{q_1}{c_1} \theta_1^{-i}}{\frac{q_1}{c_1} + \sum_{k \neq 1} \frac{q_k}{c_k} \sqrt{\frac{\theta_k^{-i}}{\theta_1^{-i}}}} \tag{10}$$

which yields the bid for the first job for the ith agent as a function of θ_k^{-i} for $k = 1, \ldots, K^i$, assuming that e is large enough to make the expression in Equation 10 positive. Thus, given any set of policies by other agents in the network, the ith agent optimizes its performance by following the bidding strategy defined by Equation 10. The values of θ_k^{-i} represent the total payments of other agents to the kth resource and serve as indicators for congestion or demand. We describe how an agent generates $\widehat{\Theta}^{-i} := [\hat{\theta}_2^{-i}, \ldots, \hat{\theta}_K^{-i}]$, the vector of estimated values of θ_k^{-i}, in Section 6, given that the agent may not have yet determined which resources it will use.

We streamline Equation 10 by using place-holder variables and reinstall the i superscript for use in future sections:

$$u_1^i = f_i(\theta_1^{-i}, \widehat{\Theta}^{-i}) := \frac{\alpha^i - \beta^i \theta_1^{-i}}{\beta^i + \frac{\gamma^i}{\sqrt{\theta_1^{-i}}}} \tag{11}$$

where

$$\alpha^i := e^i - \sum_{k \neq 1} \frac{q_k^i}{c_k^i} \hat{\theta}_k^{-i}, \tag{12}$$

$$\beta^i := \frac{q_1^i}{c_1^i}, \tag{13}$$

$$\gamma^i := \sum_{k \neq 1} \frac{q_k^i}{c_k^i} \sqrt{\hat{\theta}_k^{-i}}. \tag{14}$$

Intuitively, α^i represents the estimate of the money available for the current job. If α^i is less than zero, the agent cannot afford to purchase service under the current state of the network. If $\alpha^i \leq 0$, f_i will return a negative value, but we require that the bids be nonnegative. Thus the agent can only submit a bid if $\alpha^i > 0$. We also have $\beta^i > 0$, $\theta_k^{-i} > 0$, and $\gamma^i \geq 0$ with equality only if the agent has one job. The function $f_i(\theta_1^{-i}, \widehat{\Theta}^{-i})$ will return a positive value, if and only if, there is a feasible solution. To capture the possibility that the ith agent will choose not to bid, under

certain network conditions (which corresponds to $f_i(\theta_1^{-i}, \widehat{\Theta}^{-i}) \leq 0$), we express the bidding strategy as follows:

$$u_1^i = \max\{0, f_i(\theta_1^{-i}, \widehat{\Theta}^i)\} \tag{15}$$

4. Multiple-agent solution

An agent's bidding function in the form of Equation 15 returns the agent's optimal payment given the actions of the other agents in the network. In this section, we describe how to find an allocation that satisfies each agents' bidding strategy. Such an allocation defines a Nash equilibrium among interested agents—an allocation from which no agent can gain an advantage by unilaterally changing its actions [1]. We generate a set of bids, characterized by the expression in Equation 15, that yields a Nash equilibrium with respect to the policies of the N agents at a host:

$$\left\{u_1^i = \max\left\{0, f_i(\theta_1^{-i}, \widehat{\Theta}^{-i})\right\}\right\}_{i=1}^{N}. \tag{16}$$

We assume, without loss of generality, that the agents present are all completing their first tasks. They may, however, have itineraries of different lengths. Our analysis holds for the case when agents submit positive real-valued triples, $(\alpha^i, \beta^i, \gamma^i)$, describing their bid functions in the form of Equation 11, but easily generalizes to a broader class of bid functions that we define in Section 5.

The server collects agents' bidding functions and calculates the payments for each agent. To facilitate this computation, we translate each agent's bid function domain from θ_1^{-i} to a domain common to all agents, θ_1, to reduce our search space. Recall that we defined $\theta_1^{-i} := \theta_1 - u_1^i$ and that $\theta_1 = \sum_i^N u_1^i$. We modify the policies in Equation 16 to get an implicit relation between u_1^i and θ_1:

$$\left\{u_1^i = \max\left\{0, f_i(\theta_1 - u_1^i, \widehat{\Theta}^{-i})\right\}\right\}_{i=1}^{N}. \tag{17}$$

From Equation 17, we obtain an explicit function $g_i(\theta_1, \widehat{\Theta}^{-i}) : \theta_1 \times \widehat{\Theta}^{-i} \rightarrow u_1^i$. The ratio α^i/β^i represents agent i's tolerance for competition for the resource. Outside the range $\theta_1 \in (0, \alpha^i/\beta^i)$, $g_i(\theta_1, \widehat{\Theta}^{-i})$ takes the value of 0. Figure 2 illustrates how $f_i(\theta_1 - u_1^i, \widehat{\Theta}^{-i})$ shifts as θ_1 varies. The intersection of the line with slope 1 and a curve $f_i(\theta_1 - u_1^i, \widehat{\Theta}^{-i})$ for a particular value of θ_1^{-i}, represents the only stable solution among the set of agents for that level of congestion.

We now derive $g_i(\theta_1, \widehat{\Theta}^{-i})$. Substituting $\theta_1 - u_1^i$ for θ_1^{-i} in Equation 11, in the range, $\theta_1 \in (0, \alpha^i/\beta^i)$, we have:

$$u_1^i = \frac{\alpha^i - \beta^i(\theta_1 - u_1^i)}{\beta^i + \frac{\gamma^i}{\sqrt{\theta_1 - u_1^i}}}, \tag{18}$$

which leads to a quadratic equation in u_1^i. Dropping the i superscript, we have:

$$\gamma^2 u_1^2 + (\alpha - \beta\theta_1)^2 u_1 - (\alpha - \beta\theta_1)^2 \theta_1 = 0. \tag{19}$$

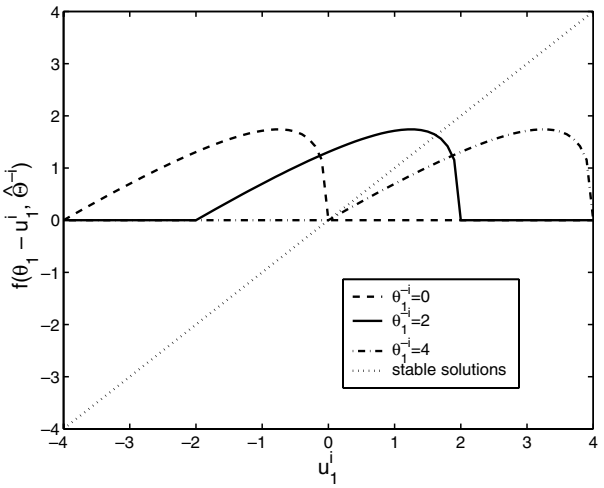

Figure 2. Behavior of $\max(0, f_i(\theta_1 - u_1^i), \widehat{\Theta}^{-i})$ for $\alpha^i/\beta^i = 4$.

Taking the positive root of the equation with respect to u_1, we have $u_1 = g(\theta_1, \widehat{\Theta}^{-i})$ where

$$g(\theta_1, \widehat{\Theta}^{-i}) = \frac{(\alpha - \beta\theta_1)^2}{2\gamma^2}\left(-1 + \sqrt{1 + \frac{4\gamma^2\theta_1}{(\alpha - \beta\theta_1)^2}}\right), \tag{20}$$

when $\theta_1 \in (0, \alpha/\beta)$, and $u_1 = 0$ otherwise.

The function g is continuous and zero at $\theta_1 = 0$ and $\theta_1 = \alpha/\beta$. Thus, $g(\theta_1, \widehat{\Theta}^{-i})$ is a continuous function of θ_1. We also note that on $\theta_1 \in (0, \alpha/\beta)$,

$$\frac{\partial g}{\partial \theta_1} = \frac{2\beta(\alpha - \beta\theta_1)}{2\gamma^2} + \frac{-2\beta(\alpha - \beta\theta_1)^2 + 2\gamma^2(\alpha - \beta\theta_1) - 4\beta\gamma^2\theta_1}{2\gamma^2\sqrt{(\alpha - \beta\theta_1)^2 + 4\gamma^2\theta_1}}, \tag{21}$$

and when $\theta_1 = 0^+$, we have

$$\left.\frac{\partial g}{\partial \theta_1}\right|_{\theta_1 = 0^+} = \frac{1}{2\gamma^2}\left[2\beta\alpha + \frac{-2\beta\alpha^2 + 2\gamma^2\alpha}{\sqrt{\alpha^2}}\right] = 1. \tag{22}$$

Figure 3 displays the shape of $g_i(\theta_1, \widehat{\Theta}^{-i})$, and how the agent's best response changes as an agent becomes wealthier, and as the agent's future task sizes increase. Increasing the agent's endowment allows the agent to submit a larger positive bid over a larger domain compared with the agent's initial bid. An agent with a greater future load, but with the same initial endowment, participates in a smaller set of congestion levels, and makes smaller bids than under the original budget when it does participate, since it must allocate more of its capital to later jobs.

Returning to the choice of an equilibrium bid for N agents, we seek θ_1 and $\{u_1^i\}_{i=1}^N$ to satisfy for all agents, the definition $\theta_1 = \sum_{i=1}^N u_1^i$ and Equation 17. An equivalent problem is to find a value of θ_1 such that $\sum_{i=1}^N u_1^i - \theta_1 = \sum_{i=1}^N g_i(\theta_1, \widehat{\Theta}^{-i}) - \theta_1 =: h_1(\theta_1) = 0$. From Equation 22, we know that if $N \geq 2$, $\partial h_1/\partial\theta_1|_{\theta_1 = 0^+} = -1 + \sum_{i=1}^N 1 > 0$, and thus h_1 is increasing to the right of zero and

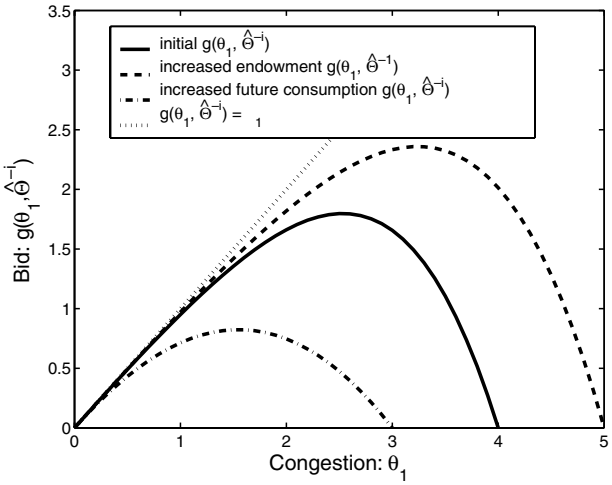

Figure 3. The comparison of $g_i(\theta_1, \widehat{\Theta}^{-i})$ for different conditions. We represent the bid for the initial itinerary with a solid curve. The dashed curve represents a new bid to complete the same set tasks, but with a larger endowment. It is scaled out compared to the original bid. The dotted-dashed line represents a bid when the itinerary has increased future consumption; the curvature is softened, and the range over which the agent participates is truncated.

$h_1(0^+) > 0$. We also know that for the nontrivial case where at least two agents have $\alpha^i > 0$, $h_1(\max_i\{\alpha^i/\beta^i\}) = -\max_i\{\alpha^i/\beta^i\} < 0$. Because h_1 is the sum of continuous functions, h_1 is continuous as well, and must be zero for some value of $\theta_1 \in (0, \max_i\{\alpha^i/\beta^i\})$. We solve for this value by using a bisection search of h_1 in the given range. We sketch a sample of $\sum_{i=1}^{N} g_i(\theta_1, \widehat{\Theta}^{-i})$ versus θ_1 in Figure 4.

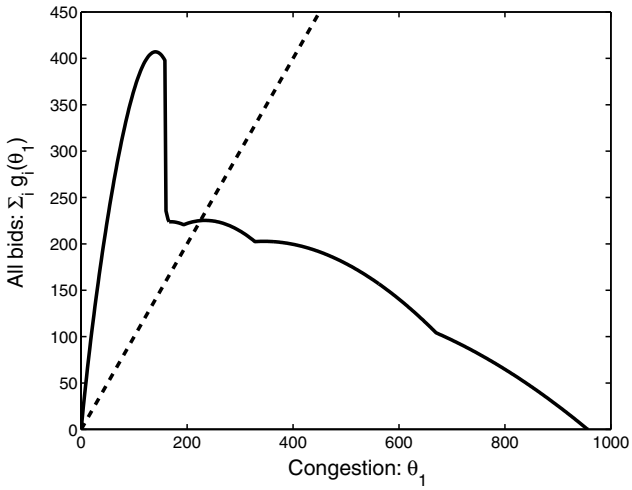

Figure 4. Sample plot of $\sum_{i=1}^{N} g_i(\theta_1, \widehat{\Theta}^{-i})$ versus θ_1 for 16 agents. Equilibrium occurs at $\sum_{i=1}^{N} \times g_i(\theta_1, \hat{\Theta}^{-i}) = \theta_1$ (i.e., $h_1(\theta_1) = 0$), which we show at the intersection of the dotted line and the plotted curve.

If an agent has only one job left to complete, $u_1 = \theta_1$ for $\theta_1 \in (0, ec/q)$ is its optimal policy, which is equivalent to having $\gamma = 0$. In this case, $g_i(\theta_1, \widehat{\Theta}^{-i})$ is discontinuous. By applying L'Hôpital's rule to Equation 20, however, we see that

$$\begin{aligned}\lim_{\gamma\to 0^+} g &= \lim_{\gamma\to 0^+} \frac{\frac{1}{2}\frac{8\theta_1\gamma}{(\alpha-\beta\theta_1)^2}(\alpha-\beta\theta_1)^2}{4\gamma\sqrt{1+\frac{4\gamma^2\theta_1}{(\alpha-\beta\theta_1)^2}}} \\ &= \lim_{\gamma\to 0^+} \frac{\theta_1}{\sqrt{1+\frac{4\gamma^2\theta_1}{(\alpha-\beta\theta_1)^2}}} \\ &= \theta_1.\end{aligned}$$

If we require agents with only one job to submit bid functions with $\gamma > 0$, agents may approximate their optimal solutions to arbitrary precision through specifying a small value for γ, and still preserve the structure of $g_i(\theta_1, \widehat{\Theta}^{-i})$ to yield a solution. Since $g_i(\theta_1, \widehat{\Theta}^{-i})$ defines an agent's best response for all values of θ_1, we have a Nash-equilibrium allocation among agents bidding for a common resource under our scheduling mechanism.

5. Uniqueness

We now show that the equilibrium (i.e., $h(\theta_1) = 0$) in the previous section is unique when there are more than two agents bidding at the same host. Let $O_i = (0, \alpha^i/\beta^i)$ be indexed such that $O_1 \supset O_2 \supset \cdots \supset O_N$ (i.e., $\alpha^1/\beta^1 > \alpha^2/\beta^2 > \cdots > \alpha^N/\beta^N$), where N is the number of agents at a server. We have already shown that a Nash equilibrium characterized by $h_1(\theta_1) = 0$ has at least one solution on $O = \bigcup_{i=1}^N O_i = (0, \max_i\{\alpha^i/\beta^i\}) = O_1$. We now further strengthen that result. Let us define $h_1^n(\theta_1)$ as follows:

$$h_1^n(\theta_1) = \sum_{i=1}^{n} g_i(\theta_1, \widehat{\Theta}^{-i}) - \theta_1. \tag{23}$$

Theorem 1 *There is exactly one solution on O where $h_1^N(\theta_1) = 0$.*

To prove Theorem 1, we must first prove some other results. In Appendix A, we show that $(\partial^2 g_i/\partial\theta_1^2) < 0$ on O_i. From the definition of h_1^n in Equation 23 and the definition of the indices, i and n, it can be seen that

$$\frac{\partial^2 g_i}{\partial\theta_1^2} < 0 \quad \text{on } O_i \;\forall i \Rightarrow \frac{\partial^2 h_1^n}{\partial\theta_1^2} < 0 \quad \text{on } O_n, \tag{24}$$

$$\left.\frac{\partial g_i}{\partial\theta_1}\right|_{\theta_1=0^+} = 1 \quad \forall i \Rightarrow \left.\frac{\partial h_1^n}{\partial\theta_1}\right|_{\theta_1=0^+} = n-1, \tag{25}$$

$$g_i(0, \widehat{\Theta}^{-i}) = 0 \quad \forall i \Rightarrow h_1^n(0) = 0. \tag{26}$$

Also, h_1^n is a continuous function of θ_1.

Lemma 1 *If a function,* $h(x)$, *is twice continuously differentiable on* $[r, s]$, $(\partial^2 h/\partial x^2) < 0$ *on* (r, s), $h(r) > 0$, *and* $h(s) < 0$, *then there exists a unique point* $x_0 \in (r, s)$ *s.t.* $h(x_0) = 0$.

Proof: We prove this lemma by contradiction. The Intermediate Value Theorem states that there is at least one value $x_0 \in (r, s)$ s.t. $h(x_0) = 0$. Because $(\partial^2 h/\partial x^2) < 0$ on (r, s), we know that h is strictly concave on (r, s), i.e.,

$$ah(x) + (1 - a)h(y) < h(ax + (1 - a)y), \tag{27}$$

for $a \in (0, 1)$ and $x, y \in (r, s)$. Suppose there are two points that satisfy $h(x) = 0$, say $x_1, x_2 \in (r, s)$ where $x_1 < x_2$. Again, using the Intermediate Value Theorem, we can show that $\exists\, r_0 \in (r, x_1) \subset (r, s)$ s.t. $h(r_0) > 0$. Then, we have $ah(r_0) + (1 - a)h(x_2) < h(ar_0 + (1 - a)x_2)$ which implies $ah(r_0) < h(ar_0 + (1 - a)x_2)$. If $a = (x_2 - x_1/x_2 - r_0) \in (0, 1)$, then we have $ah(r_0) < h(x_1) = 0$, which is a contradiction since $a > 0$. Thus, there can be at most one point where $h(x) = 0$. $\Diamond$

Proof of Theorem 1: When $n = 1$, $(\partial h_1^1/\partial \theta_1)|_{\theta_1 = 0^+} = 0$, and $(\partial^2 h_1^1/\partial \theta_1^2) < 0$ on O_1, which implies that $h_1^1(\theta_1) < 0$ on O_1. When $n = 2$, $(\partial h_1^2/\partial \theta_1)|_{\theta_1 = 0^+} = 1$ and $h_1^2(0) = 0$, thus $h_1^2(0^+) > 0$. Also, $h_1^2(\alpha^2/\beta^2) = h_1^1(\alpha^2/\beta^2) < 0$ and $(\partial^2 h_1^2/\partial \theta_1^2) < 0$ on O_2. Applying Lemma 1, we find that there is a unique point θ_0 s.t. $h_1^2(\theta_0) = 0$ on O_2. But, $h_1^2(\theta_1) = h_1^1(\theta_1) < 0$ on $O_1 \cap O_2^c$; thus θ_0 is a unique point where $h_1^2(\theta_0) = 0$ on O_1.

We show uniqueness through an inductive argument. Assume that there is a unique point $\theta_0 < \alpha^i/\beta^i$ on O_1, where $h_1^i(\theta_0) = 0$. Also assume $(\partial^2 h_1^i/\partial \theta_1^2) < 0$ on O_i and $h_1^i(\theta_1) < 0$ on $O_1 \cap O_i^c$. Along with the continuity of h_1^i, the previous result implies the following:

$$h_1^i(\theta_1) \begin{cases} > 0 & \theta_1 < \theta_0 \\ = 0 & \theta_1 = \theta_0 \\ < 0 & \theta_1 > \theta_0 \end{cases}. \tag{28}$$

Rewriting Equation 23, we have $h_1^{i+1} = h_1^i + g_{i+1}$. There are two cases to consider:

Case 1. If $(\alpha^{i+1}/\beta^{i+1}) \leq \theta_0$, then Equation 28 is satisfied for h_1^{i+1}, because $g_{i+1}(\theta_1) = 0$ for $\theta_1 \geq \theta_0 \geq (\alpha^{i+1}/\beta^{i+1})$ and $g_{i+1}(\theta_1) \geq 0$ for $\theta_1 \leq (\alpha^{i+1}/\beta^{i+1})$. Thus, there is a unique point θ_0 where $h_1^{i+1}(\theta_0) = 0$ on O_1.

Case 2. If $\theta_0 < (\alpha^{i+1}/\beta^{i+1}) < (\alpha^i/\beta^i)$, then by Equation 28, $h_1^{i+1}(\alpha^{i+1}/\beta^{i+1}) = h_1^i(\alpha^{i+1}/\beta^{i+1}) < 0$. We also know $h_1^{i+1}(0^+) > 0$, because $h_1^{i+1}(0) = 0$ and $(\partial h_1^{i+1}/\partial \theta_1)|_{\theta_1 = 0^+} = i > 0$. Since $(\partial^2 h_1^{i+1}/\partial \theta_1^2) < 0$ on O_{i+1}, we can apply Lemma 1 and arrive at the result that there is a unique point $\hat{\theta}_0$ on O_{i+1}, where $h_1^{i+1}(\hat{\theta}_0) = 0$. But since $g_{i+1}(\theta_1) = 0$ for $\theta_1 > (\alpha^{i+1}/\beta^{i+1})$, we have that on $O_1 \cap O_{i+1}$, $h_1^{i+1}(\theta_1) = h_1^i(\theta_1) < 0$. Thus, we have a unique point $\hat{\theta}_0$ where $h_1^{i+1}(\hat{\theta}_0) = 0$ on O_1. $\Diamond$

We have shown that a unique Nash equilibrium exists when agents' bidding functions are in the form of Equation 20, but we note that the results hold for any set of functions $g(\theta, \widehat{\Theta}^{-i})$, where $g(\theta, \widehat{\Theta}^{-i}) > 0$ on some set $(0, b)$, and is zero elsewhere, and further, $g(\theta, \widehat{\Theta}^{-i})$ is concave and twice differentiable on $[0, b]$.

6. Implementation in a network

In this section, we discuss several issues that concern calculation of an agent's bid. Implicit in the calculation are both the agent's resource choices, and its forecast of network conditions. We explore the possibility of other bidding strategies and conclude the section with a discussion of the merits of our function-submission bidding allocation policy.

6.1. Estimation

In our approach, a resource independently partitions itself as demand changes. Every time an agent arrives at, or finishes with, a resource, each agent that requires access submits an updated bidding function characterized by the real-valued positive triple, (α, β, γ). The parameters α and γ depend on $\hat{\Theta}^{-i}$, the agent's beliefs about future demand for resources on its itinerary. These estimates allow an agent to budget expenditures for the current task.

Two issues that affect the estimations are the particular resources that the agent chooses to access, and the congestion of resources used later in the itinerary. A greedy algorithm, which we present in Section 7, chooses an agent's next resource. The algorithm defers selection of resources until execution of the corresponding task. The agent must have an idea of the quality of future resources, it will choose to budget expenditures for the immediate task, however. For this purpose, an agent in our model uses the mean of the alternative resources' capacities to estimate the capacity of the resource it will use.

Additionally, the budgeting algorithm requires congestion estimates. The agent forecasts these through calculation of the mean total cash spent by other agents at the resource alternatives over time. In Section 7.2 we examine the effect of congestion estimation errors that stem from network delay. We find that the effects on agents' performance are limited.

6.2. Alternative bidding strategies

Once agents submit their bidding functions, the resource partitions itself based on the Nash-equilibrium allocation. Thus, each agent will be satisfied with the amount and cost of the resource that it receives, as the Nash equilibrium ensures that the allocation represents a point on the optimal response function of each agent for that resource at that time given its current beliefs. Each agent continues under this allocation until its job is complete or the resource state changes, i.e., another

agent arrives or leaves the resource. Agents that pay nothing under the allocation are ignored until the state changes again. When an agent completes its task at a resource, it moves to the next resource in its itinerary and submits a new bidding function with updated demand forecasts.

It is possible for an agent to "cheat" by submitting a bid that does not represent its true beliefs. An agent might hope that an altered equilibrium might expedite itinerary completion. In other applications, agents can gain from such strategic optimization in double auctions [13]. A difference between our allocation problem and the double-auction research, is that in a double auction there are trade opportunities while the market searches for equilibrium.

Trade in our system occurs only at equilibrium, so strategic gaming is more difficult, since an agent has only one chance to manipulate the equilibrium. Because agents observe only the resulting equilibrium, as opposed to the search for an equilibrium, there is less information to use and fewer opportunities to steer the equilibrium in a favorable direction. There are other complications in optimization. One must consider the feedback of one agent's actions from its competition's actions. The effects of indirect interaction among agents on their beliefs, and bidding behavior is an interesting topic for future research.

Submitting functions as bids ensures an equilibrium allocation that a resource owner can compute quickly. An alternative solution might be to iteratively collect bids, and let each allocation drive the recalculation of the next iteration. This sequence does not guarantee a convergence, however. Our policy allows an agent to calculate and express its actions for every state, and results in efficient trade even in the presence of fluctuating demand.

7. Simulation

In this section, we present results of simulating our resource-allocation policy from Section 4. We show that agent performance is highly correlated with endowment; that when our system is overloaded, poorer agents are ignored to maintain higher priority agents' performance; our algorithm is robust to an agent's errors in job-size estimation; and that there is a empirical structure to bidding behavior that will aid us in future research to predict server loads, and guarantee service to agents.

We ran our simulations under the Swarm simulation system [17]. We created agents at a Poisson rate, each with an exponentially distributed number of jobs to complete. In our simulations, we used two different distributions for job size: exponential and Pareto. Both distributions are commonly used to model job sizes, with Pareto having the more sporadic distribution. Each agent's start location and task types were uniformly distributed. We modeled ρ^i, an agent's endowment size relative to the sum of its job sizes, as a positive truncated Gaussian random variable. This parameter expresses an agent's owner's preference that the agent completes tasks quickly.

All of our simulations used a network of 100 hosts where each host offered one of eight computational services. The service that a host offered was picked uniformly at simulation initialization. Host capacity was determined by a positive

truncated Gaussian random variable with positive mean. We chose this distribution for no other reason than to make the process of host selection more important, and to show that our expenditure planning process works with heterogeneous host capability. In the simulation, each host published its fixed capacity and immediate level of congestion, θ_k.

The hosts were connected with a network whose topology we generated with GT-ITM [6]. In GT-ITM, a network is built from a hierarchical system of transit domains connecting stub domains. The user specifies the number and average size of domains in nodes (hosts), and the probability that nodes are connected within the domain. Our networks had two levels of transit domains connecting another set of stub domains. The network delay incurred in an agent moving between sites was chosen at system initialization. We chose job sizes to be large enough so that network transfer did not dominate an agent's decision of which hosts to visit, since we were interested in the effectiveness of our expenditure-planning algorithm and resource-allocation policy.

Once created, an agent must formulate a route. In the simulation, each agent chose their route incrementally, by choosing a host for each task after completing the previous task. For the purpose of expenditure planning, for all but the next immediate host choice, agents planned to visit hosts of average capacity c_k, and average congestion θ_k, among hosts offering the service required for the kth task. Each agent chose the next site to be the one that minimized the sum of network-transfer and execution times for the next hop, assuming that the bidding level, θ_k^{-i}, would not change. Thus, our routing algorithm was greedy and naive. We sketch its operation in Algorithm 1.

In the simulation, an agent commits to finishing its current job at the host to which it jumps. To finish its job, an agent submits a bid function defined with parameters $\alpha^i, \beta^i, \gamma^i$ defined in Equations 12–14. The host uses bids from agents to form the bid-response function, $g(\theta, \widehat{\Theta}^{-i})$, and uses a bisection search to find the bidding level where $\theta = \sum_{i=1}^{N_k} g(\theta, \widehat{\Theta}^{-i})$. This search is conducted every time an agent arrives or departs the host. Algorithm 2 sketches the operation.

We compared our game-theoretic resource-allocation method with three other resource-allocation policies: equally-shared, first-come-first-served (FCFS), and shortest-remaining-processing-time (SRPT). Under the shared policy, all agents at a site compute at an equal rate. The shared policy is similar to what is currently used in many existing mobile-agent systems, so it serves as a good comparison for performance.

The FCFS policy allocates all of a host's capacity to the earliest arriving agent, while the SRPT policy services the agent with the shortest computation remaining. We examined SRPT because it minimizes the average agent job-completion time, and serves as a lower bound on the metric. The SRPT policy, however, has another side effect: it prioritizes jobs by their size, so users have incentive to understate the size of their jobs and the resulting allocation would resemble FCFS. In our simulations using SRPT, we assumed that agents' job sizes were known to the host.

An agent operating under the shared-allocation policy chose its next host to be the one that minimized the sum of network-transfer, and the execution times of the

Algorithm 1 `Choose Next Site for Agent` i

$t_{min} := \infty$; $nextHost := \emptyset$
for all hosts j offering service next in itinerary **do**
  $t_j :=$ [`transferLatency to:` j `from:` $currentHost$]
    $+q_j g(\theta_j^{-i}, \hat{\Theta}^{-i})/(c_j * (\theta_j^{-i} + g(\theta_j^{-i}, \hat{\Theta}^{-i})))$
  if $t_{min} > t_j$ **then**
    $t_{min} := t_j$; $nextHost := j$
  end if
end for
return $nextHost$

Algorithm 2 `Allocate Resources for Host` k

while true do
  $t :=$ time since last arrival/departure
  for all agents i **do**
    deduct $tg_i(\theta, \hat{\Theta}^{-i})$ from agent i's endowment
  end for
  add arriving agent or remove departing agent
  for all agents i **do**
    query agent i for α, β, and γ to build $g_i(\theta, \hat{\Theta}^{-i})$
  end for
  search for $\theta = \sum_{i=1}^{N} g_i(\theta, \hat{\Theta}^{-i})$ in $(0, \max_i(\alpha_i/\beta_i))$
  for all agents i **do**
    $v_j^i := c_j g_i(\theta, \hat{\Theta}^{-i})/\theta$
  end for
end while

next task given the computational share that an additional agent would receive at the host. Under the FCFS and SRPT policies, each agent chose its next site to be one that minimized the sum of the network-transfer time to the host, and the host's ideal job completion time weighted for the number of agents currently at the host. For agents to plan their itineraries using each of the policies, sites published the number of agents visiting them.

We measured an agent's performance by comparing the actual time taken by the agent, with performance that it would have achieved in a network with zero congestion. This idealized measurement is a shortest-path computation from the start location that visits hosts that offer the appropriate services. In the calculation, the distance between any two hosts is the sum of the network-transfer time, and q_k/c_k, the time an agent would take to complete a job at the second host without any competition.

We ran three sets of experiments. First we verified that our game-theoretic resource-allocation policy prioritizes agents by endowment. We compared the performance agents achieved under the policy with the mean performance achieved by agents under the other policies. We also examined how over-constrained

resources are allocated. Second, each of the policies requires agents to compute routes based upon network state, so we also investigated the effect of network delay on an agent's ability to plan its itinerary and budget. Finally, both the SRPT and the game-theoretic allocation policies rely on agents having knowledge of their job sizes. In reality, there will be some error in an agent's estimation of an agent's computational requirements, so we looked at the effect of job-size estimation error on agents' performance.

7.1. Effectiveness

To test the effectiveness of the game-theoretic resource-allocation policy, we compared agents' priorities, expressed by ρ^i, with their performance. After the network reached a steady state, we designated 7% of the agents injected into the system as test agents. The test agents had identical task-type sequences and a common start host, but they had differing priorities, expressed by ρ^i, spanning two standard deviations, σ, around the mean priority, μ, and differing task sizes.

Figure 5 shows how endowment relative to task size affects agent performance in one experiment and how agents performed in separate simulations using the shared, FCFS, and SRPT policies. The data labeled "GT" show the performance of agents operating under our game-theoretic allocation policy. We plot the mean and standard deviations of agents' performance versus their priority. We also plot the mean performance of all agents operating under the shared, SRPT, and FCFS allocation policies in separate simulations. In the experiment, agents' job sizes had

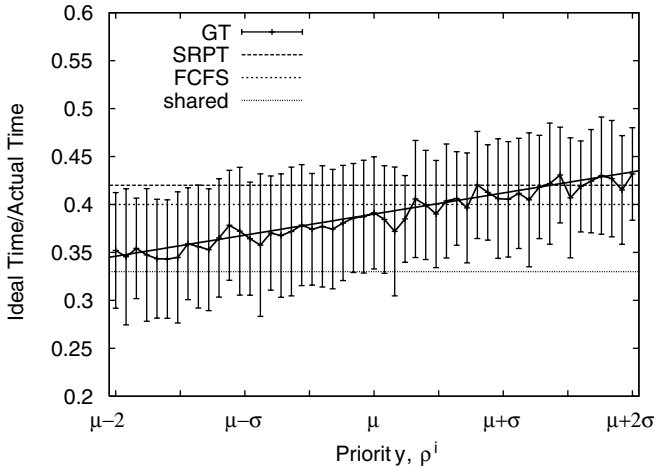

Figure 5. Priority versus ideal time relative to actual time. For the game-theoretic (GT) approach, we plot the observed means in heavy block lines; standard deviations with error bars for agents at each priority two standard deviations, σ, around the mean priority, μ; and a linear fit of the observed means. We also plot the mean performance of agents under the shared, SRPT, and FCFS resource-allocation policies.

a Pareto distribution, but we achieved similar results when agents' job sizes were exponentially distributed.

We plot the mean of the ratios of the ideal noncongested itinerary completion time, and the actual itinerary completion time for agents. A higher number indicates better performance. In the experiment, agents with higher endowments performed better on average than those with lower endowments. The mean performance of agents using the shared policy was 0.33, while agents across all priorities under the game-theoretic policy achieved a mean performance of 0.39—an improvement of 18%. There was a cost to prioritizing agents, however. Agents under the FCFS policy performed slightly better (0.40) than under the game-theoretic policy, and the SRPT experiment showed that ideal performance was 8% better than the mean performance of agents across all priorities operating under the game-theoretic policy. Neither the SRPT, nor the FCFS policies prioritize agents, however.

One might associate high variance with market-based techniques, but in the plotted simulations, agents using the game-theoretic policy experienced about the same amount of performance variance as did agents using the shared allocation policy. In the experiment plotted in Figure 5, the mean standard deviation of test agents' performance across all priorities was 0.065, while agents using the shared resource-allocation policy had a performance standard deviation of 0.068. Agents operating under the FCFS policy experienced even higher variance.

The variance in performance in the game-theoretic simulations depended on the level of congestion. As congestion increased, endowment became a much stronger factor in determining an agent's performance and the variance of the performance measure increased.

Figure 6 shows how the system prioritized agents when requests exceeded system capacity. We observed that agents with lower than mean priority were not able to complete their itineraries at all. The resulting equilibrium allocation pushed prices beyond the range in which poorer agents' bids returned positive payments. Among

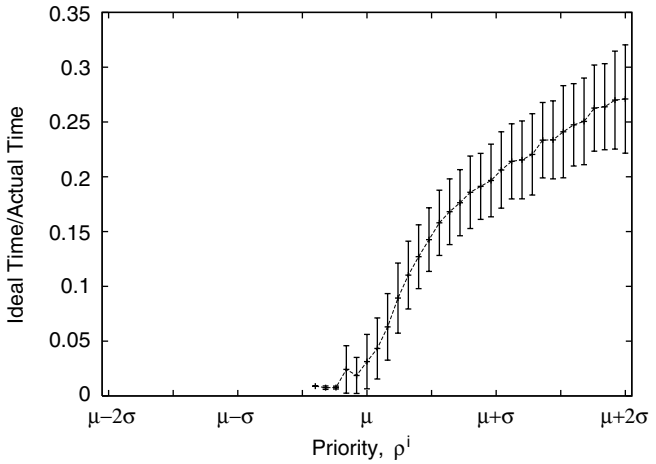

Figure 6. Priority versus ideal time relative to actual time when agent requests exceeded capacity. We plot observations two standard deviations, σ, around the mean priority, μ.

agents with higher than average priority, there was a strong correlation of intended priority and performance. Again, we achieved similar results when agents had either Pareto or exponentially distributed job sizes.

Using the other resource-allocation policies, it is not possible to reach a steady state when agents' requests exceed capacity. The requests that are completed are completed more and more slowly as time progresses. The lack of a steady state in the shared, FCFS, and SRPT policies illuminates another feature of the game-theoretic policy: because agents are prioritized, the system can decide which requests to postpone and still provide reasonable service to higher priority requests. The SRPT policy prioritizes agents, but in an uncontrollable manner, and one that is not meaningful when the system is overloaded.

7.2. Network delay

We also examined the importance of timely information to our expenditure planning algorithm. We ran experiments varying the latency incurred in agents jumping from one site to another. In the model, agents have information on the immediate state of the world, but there is a delay in acting on the information in the form of the time required to move to another site. So by varying the agent-transport latency, we effectively aged the agents' load information.

Figure 7 shows the results of four experiments that used the game-theoretic resource-allocation policy with intra-domain transfer times of one, two, four, and eight time units. Inter-domain transfer times are three, six, 12, and 24 time units. The experiments all used Pareto job-size distributions, and increased job sizes relative to server capacity to increase the granularity of network transfer times. We saw that recent information is valuable to agents and, as their load information

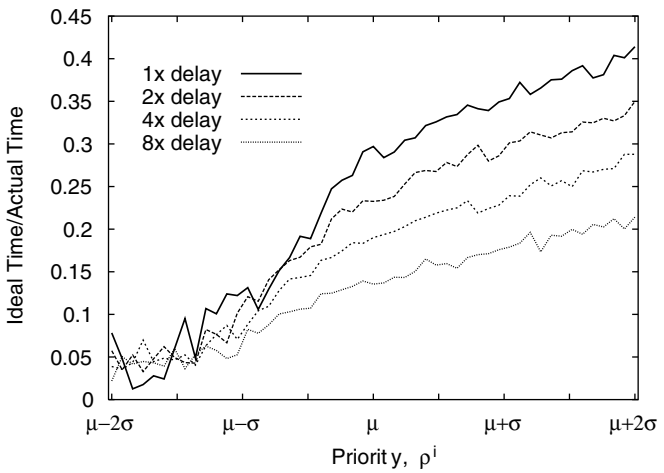

Figure 7. We plot the effectiveness of prioritization of the allocation policy at different network delays and levels of endowment. We plot observations two standard deviations, σ, around the mean priority, μ.

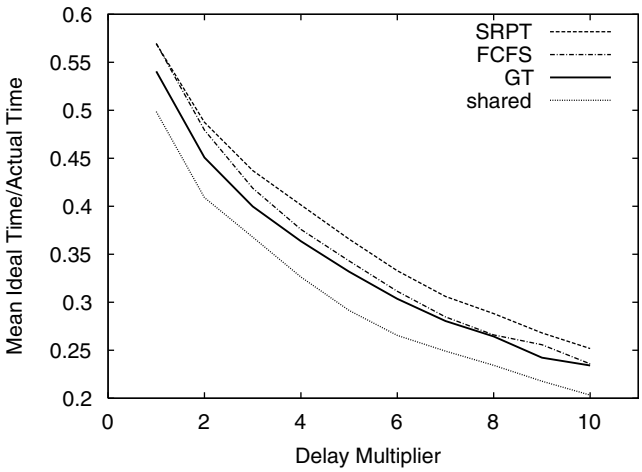

Figure 8. The mean performance of agents operating in various allocation policies versus network delay.

aged, ideal/actual performance ratio gradually decayed. The game-theoretic allocation policy maintained priority stratification as network delay increased.

The degradation of performance was not unique to the game-theoretic policy, however. In Figure 8 we compare the mean performance of agents at all priorities operating under the game-theoretic policy to the mean performance of agents operating under the other resource-allocation policies at different network delays. Under all of the policies, we observed that agents' performance decayed gradually as they used more dated information.

7.3. Estimation error

Both the game-theoretic and SRPT policies rely on each agent knowing the number of instructions involved in its computation before the calculation is commenced. In practice, the number of instructions will not be known to agents ahead of time. Furthermore, in most architectures, different instructions require varying amounts of time to complete. For these two reasons, we investigated the effect of error in agents' job-size estimation.

We ran several experiments under the game-theoretic and SRPT allocation policies, and varied agents' accuracy in predicting their job sizes. We modeled an agent's ability to estimate its job sizes as a truncated Gaussian random variable with mean one. An agent's estimation of its job size was the product of the random variable and the true job size. The standard deviation of the random variable determined the error in the estimation.

Figure 9 shows how agents' performance changed as we varied all agents' job-size estimation error from perfectly accurate to situations where the standard deviation of agents' job-size estimates was five times job size. Within each experiment, all agents used a common error distribution function to generate an imperfect job size measurement. We observed a modest reduction in agents' performance as error

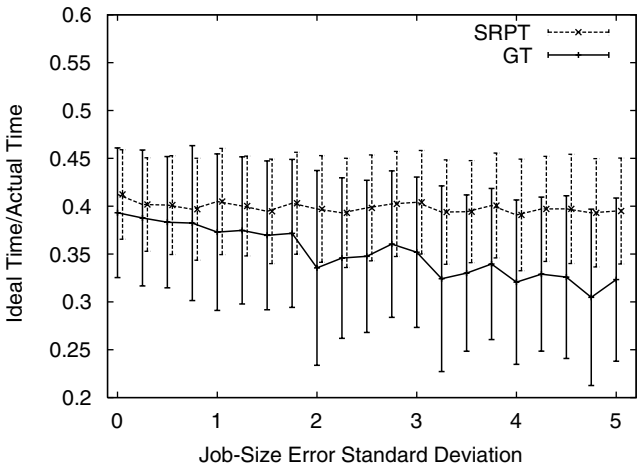

Figure 9. Agent performance versus the standard deviation of job-size error.

increases. Agents experienced approximately a 3% decrease in performance for every additional multiple in the standard deviation of their job-size estimation error.

7.4. Bidding patterns

Information concerning the behavior of bid totals, θ, will be useful for more sophisticated planning algorithms. We plot histograms of the logarithm of positive bid totals with their best-fit log-normal distributions in Figures 10 and 11 for experiments that

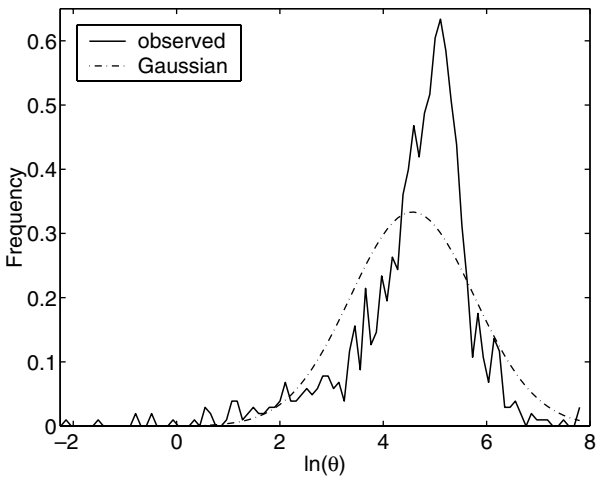

Figure 10. A histogram of bid sum on a log scale, ln(θ), observed at a resource where agents arrived with exponentially distributed job sizes. For comparison, we plot the best-fit log-normal distribution.

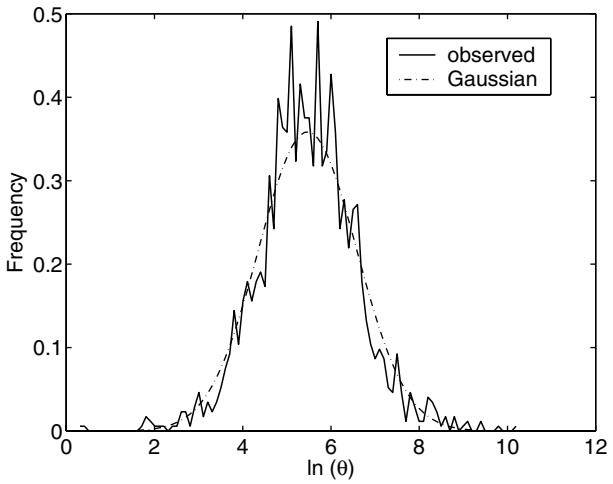

Figure 11. A histogram of bid sum on a log scale, ln(θ), observed at a server where agents arrived with Pareto distributed job sizes. For comparison, we plot the best-fit log-normal distribution.

used exponentially and Pareto distributed job sizes, respectively. Both experiments gave similar results. The bids in the Pareto data were close to log-normally distributed. The observed cumulative distribution function deviated from the corresponding log-normal cumulative distribution by no more than 0.03. The experiment that used exponentially distributed job sizes produced a bid total distribution that visually resembles a log-normal distribution, but it was skewed away from the origin and produced a poor fit.

The ability to fit bid totals to a known distribution will aid in constructing predictors for server loads that will allow us to relax assumptions on agents' knowledge of the state of the network, aid in constructing better expenditure planners, and give hosts insight that allow them to issue efficient resource-consumption reservations.

8. Related work

Our earlier work relies on agents submitting demand functions derived to optimize Cobb-Douglas utility functions subject to their budget constraints [5]. Sellers compute equilibrium prices given their preferences for consuming their own computational resources and buyers' demand functions. More recently, we investigated seller-driven markets, where servers supply price curves increasing with rates of computation from which agents choose their rates of service [18]. Both works treat agents' consumption habits as a local problem. As such, expenditure planning is only indirectly handled through agents' preferences for savings, and there is no guarantee of agents' ability to complete itineraries within preset time limits.

We are not the only group to promote the use of markets in mobile-agent systems. The Geneva Messengers project [26] applies market ideas to allocate CPU usage, and memory to visiting *messengers*, lightweight mobile programs implemented in a Postscript-like language. Host sites heuristically set prices by examining the amount of resources requested by the present messengers.

Telescript [28] supports a fault-tolerance and security measure where agents carry "permits" to access specific resources. A permit's power diminishes over an agent's lifetime, thus limiting the agent's lifetime. A permit for one resource is not easily converted to a permit for another resource. A more general policy would be for hosts to issue a common permit, in the form of a verifiable electronic currency.

POPCORN [23] is a distributed framework where users submit "computelets" to a computational market that assigns the computelets to anonymous hosts that charge computelets for execution. The approach is intended for parallel programs where interaction among threads is limited, and computelets are single-hop programs so no expenditure planning is necessary. Computation is the sole resource regulated in POPCORN and computelets may not consider any other information other than hosts' prices and computational capabilities in choosing sites. Our system allows mobile agents to choose their hosts, so agents may weigh the value of hosts' reputations or network connections or the location of other agents.

The idea of using economics for computational-resource control dates back as far as the 1960s [25]. Spawn is perhaps the most cited work dealing with computational economic systems [27]. In Spawn, agents participate in auctions to buy processor time to run computationally intensive jobs. The pricing system pairs idle processors with jobs to improve utilization in distributed systems. Double auctions have been used to allow agents to trade climate-control resources within an office building [10]. The result is that climate control resources are more effectively allocated with energy savings of up to 10%.

A similar approach uses sequential auctions [4]. The research overcomes incentive compatibility problems by having each buyer express her preference to an agent identical to her competitor's. The agent then participates in iterated auctions until an equilibrium is found. The method can handle many traditionally difficult assignment problems, where goods may be complements or substitutes to one another. The technique is centralized and more general than what we model in that it makes no assumptions on resource values and relations, but it is also computationally more expensive.

Tatonnement is a resource-allocation method where buyers iteratively adjust purchase amounts in response to sellers' changing prices. The WALRAS algorithm [8] is a system for finding equilibrium among several markets. WALRAS assures equilibrium convergence if participants have convex utility functions, and there is gross substitutability among goods (goods are not complements for one another), but the system frequently converges to a solution, even when gross substitutability is violated.

Our allocation policy makes no connections between markets. It performs allocation on a resource-by-resource basis and allows for operation in disconnected environments, or where the cost of communication is high.

9. Summary

We describe a network where agents compete with each other for network resources. We introduce a scheduling mechanism based on each agent bidding for services, and receiving a portion of the resource proportional to its payment, with respect to the sum of all payments for that resource. We use an electronic market, in which agents are endowed with finite capital to complete a sequence of tasks, and derive a bidding policy as a function of the total sum of payments for the resource. We show that when agents submit their optimal response functions based on assuming fixed payments of other users, the resource can make an allocation that forms a unique Nash equilibrium among the agents. The equilibrium allocation is flexible and enforces the priorities dictated by the endowments. When there is heavy competition for a resource, agents with larger endowments receive larger portions of network resources. Simulations show correlation between endowment per-unit job and completion time. Our planning and allocation algorithms are no more dependent on the timeliness of information concerning host congestion than the other allocation policies that we consider. Furthermore, our experiments show that the results are not sensitive to the distribution of agents' workloads, and agents only suffer modest reductions in performance stemming from similar errors in their job-size estimations.

We compare our allocation policy with an allocation model used in many mobile-agent system implementations, in addition to SRPT and FCFS models. Simulations show that agents operating under our policy complete their itineraries faster than agents operating under a traditional shared-resource environment, with no additional variance incurred. Our policy is competitive with the FCFS policy. Compared to SRPT, the policy that minimizes average completion time, our policy provides a method of prioritizing jobs, and that prioritization typically results in mean agent performance across all endowment levels that is 92–95% of what is achieved under SRPT.

There are several areas open for future investigation. One is the user optimization problem: how to assign capital between various agent task sequences. The solution depends on the user having information about the correlation between endowment and performance. Creating tractable models that yield computable solutions to the endowment-assignment problem and consideration of concurrently-used resources are open challenges.

Another area for investigation is alternative utility functions for the agents. In our model, agents only consider execution time and users do not expect any part of the endowment to be returned. Another option is to have the agent consider both the execution time and the cost of computing in its utility function. Under such a model, the user would have an expectation that some of the endowed cash would be returned unless the network is congested. A comparison of costs and performance of agents operating under each utility function might lead to insight about the benefits of giving an agent greater latitude with capital.

At a more general level, there is a need for the development of a theory to describe the behavior of agents, and network resources acting in a decentralized

manner, as accurate estimates of load fluctuations and reactions to changes in market variables both depend on having useful models that yield tractable results.

Appendix A. Proof of Concavity of $g_i(\theta_1, \widehat{\Theta}^{-i})$

Theorem 2 *An agent's bidding function,* $g_i(\theta_1, \widehat{\Theta}^{-i})$, *is concave on the interval* $(0, \alpha^i/\beta^i)$.

For the variables $\alpha^i, \beta^i, \gamma^i$ defined in Equations 12–14, we drop the superscripts and consider the bidding function of only single agent. Let $w(x)$ be a function defined as:

$$w(x) = -x^2 + \sqrt{x^4 - bx^3 + b\alpha x^2}, \tag{29}$$

where $x \in (0, \alpha)$ and $b = (4\gamma^2/\beta)$.

Then, $g_1(\theta_1, \widehat{\Theta}^{-i}) = (1/2\gamma^2)w(\alpha - \beta\theta_1)$ for $\theta_1 \in O_1$ and

$$\partial^2 g_1/\partial\theta_1^2 = (\beta^2/2\gamma^2)(\partial^2 w/\partial x^2). \tag{30}$$

To prove the concavity of g_1 on O_1, it suffices to show that $(\partial^2 w/\partial x^2) < 0$ for $x \in (0, \alpha)$. We have

$$\partial^2 w/\partial x^2 = (2p(x)p''(x) - p'(x)^2 - 8p(x)^{\frac{3}{2}}/4p(x)^{\frac{3}{2}}), \tag{31}$$

where $p(x) = x^4 - bx^3 + b\alpha x^2$. Since $p(x) > 0$ for $x \in (0, \alpha)$, it is sufficient to show

$$v(x) = 2p(x)p''(x) - p'(x)^2 - 8p(x)^{\frac{3}{2}} < 0. \tag{32}$$

After substituting for $p(x)$ and simplifying, we get

$$\begin{aligned} v(b, x) = {} & -4\alpha b^2 x^3 + 12\alpha b x^4 + 3b^2 x^4 + 8x^6 \\ & - 12bx^5 - 8(x^4 - bx^3 + b\alpha x^2)^{\frac{3}{2}}. \end{aligned} \tag{33}$$

We note that $v(0, x) = 0 \ \forall x$. Taking the partial derivative of $v(x)$ with respect to b, we get

$$\begin{aligned} (\partial v/\partial b) &= -(\alpha - x)[6bx^3 + 12x^2\zeta(x)] - 2b\alpha x^3 \\ \zeta(x) &= \sqrt{x^4 + bx^2(\alpha - x)} - x^2, \end{aligned} \tag{34}$$

hence $\partial v/\partial b$ is negative for $x \in (0, \alpha)$ and $b > 0$.

$v(b, x) < 0$ for all $b > 0$ for $x \in (0, \alpha)$, which implies $(\partial^2 w/\partial x^2) < 0$ for $x \in (0, \alpha)$, and thus $(\partial^2 g_1/\partial\theta_1^2) < 0$ on O_1. $\diamondsuit$

Appendix B. Notation

α^i	$:= e^i - \sum_{k \neq 1} \frac{q_k^i}{c_k^i} \theta_k^{-i}$, simplifies expression for an agent's bid.
β^i	$:= \frac{q_1^i}{c_1^i}$, simplifies expression for an agent's bid.
γ^i	$:= \sum_{k \neq 1} \frac{q_k^i}{c_k^i} \sqrt{\theta_k^{-i}}$, simplifies expression for an agent's bid.
c_k^i	The computational capacity of the kth host that the ith agent visits.
e^i	The ith agent's endowment.
$f_i(\theta_j^{-i}, \widehat{\Theta}^{-i})$	The ith agent's optimal bid conditioned on all other agents' bids at the jth host sum to θ_j^{-i}, $\theta_j^{-1} \in (0, \alpha^i/\beta^i)$, and the estimate of future congestion $\widehat{\Theta}^{-i}$.
$g_i(\theta_j, \widehat{\Theta}^{-i})$	The ith agent's bid conditioned on all agents' bids at the jth host sum to θ_j, $\theta_j \in (0, \alpha^i/\beta^i)$, and the forecast of future congestion.
$h_1^k(\theta_1)$	The difference of θ_1 and the sum of $g_i(\theta_1, \widehat{\Theta}^{-i})$, where the sum is over the k agents with the largest positive bidding domains.
K^i	The number of tasks in the ith agent's itinerary.
m_k^i	The ith agent's expenditure at the kth host that it visits.
q_k^i	The size of the ith agent's kth job.
ρ^i	$:= e^i / \sum_{k=1}^{K_i} q_k^i$. The ith agent's endowment relative to its task sizes.
t_k^i	The time taken to complete the ith agent's kth task.
$\widehat{\Theta}^{-i}$	The vector of the estimates of the values of θ_k^{-i} for each task in the agent's itinerary.
θ_k^{-i}	The sum of the bids of all other agents visiting the kth host that the ith agent visits.
θ_k	The sum of the bids of all agents visiting the kth host.
u_j^i	The amount that the ith agent bids at the jth site it visits.
v_k^i	The rate at which the ith agent computes its kth job.

Acknowledgments

We would like to thank the invaluable and thorough input of several anonymous referees, especially in regards to the presentation of the system model.

This work was supported in part by Department of Defense contract MURI F49620-97-1-0382, DARPA contract F30602-98-2-0107, an EPRI/ARO contract, NSF Grant EIA-98-02068, ONR grant N00014-95-1-1204, and a grant from the Alfred P. Sloan Foundation.

References

1. T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory*. SIAM, Classics in Applied Mathematics, 1999.
2. M. Baldi, G. P. Picco, and F. Risso, "Designing a Videoconference System for Active Networks," in *Proceedings of the Second International Workshop, Mobile Agents '98*, Stuttgart: Germany, pp. 273–284, 1998.
3. A. Bieszczad, B. Pagurek, and T. White, "Mobile Agents for Network Management," *IEEE Communications Surveys*, vol. 1, pp. 2–9, 1998.
4. C. Boutilier, M. Goldszmidt, and B. Sabata, "Sequential Auctions for the Allocation of Resources with Complementarities," in *Proceedings of the International Joint Conference on Artificial Intelligence*, Stockholm: Sweden, pp. 527–523, 1999.
5. J. Bredin, D. Kotz, and D. Rus, "Utility Driven Mobile-Agent Scheduling," Technical Report PCS-TR98-331, Dartmouth College, 1998.
6. K. Calvert and E. Zegura, "GT-ITM: Georgia Tech Internetwork Topology Models," http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm/gt-itm.tar.gz, 1996.
7. A. Chavez, A. Moukas, and P. Maes, "Challenger: A Multiagent System for Distributed Resource Allocation," in *Proceedings of the First International Conference on Autonomous Agents*, ACM Press: Marina Del Ray, CA, pp. 323–331, 1997.
8. J. Q. Cheng and M. P. Wellman, "The WALRAS Algorithm: A Convergent Distributed Implementation of General Equilibrium Outcomes," *Journal of Computational Economics*, vol. 12, pp. 1–23, 1998.
9. S. H. Clearwater (ed.), *Market-Based Control*, Singapore: World Scientific, 1996.
10. S. H. Clearwater, R. Costanza, M. Dixon, and B. Schroeder, "Saving Energy Using Market-Based Control," in *Clearwater, 1996*, pp. 253–273, 1996.
11. R. Gagliano, M. Fraser, and M. Shaefer, "Auction Allocation of Computer Resources," *Communications of the ACM*, vol. 38, no. 6, pp. 88–102, 1995.
12. S. Glassman, M. Manasse, M. Abadi, P. Gauthier, and P. Sobalvarro, "The Millicent Protocol for Inexpensive Electronic Commerce," *World Wide Web Journal*, pp. 603–618, 1995.
13. J. Hu and M. P. Wellman, "Online Learning about Other Agents in a Dynamic Multiagent System," in *Proceedings of the Second International Conference on Autonomous Agents*, Minneapolis, MN, pp. 239–246, 1998.
14. D. Johansen, "Mobile Agent Applicability," in *Proceedings of the Second International Workshop, Mobile Agents '98*, Stuttgart: Germany, pp. 80–98, 1998.
15. D. Kotz and R. S. Gray, "Mobile Agents and the Future of the Internet," *ACM Operating Systems Review*, vol. 33, no. 3, pp. 7–13, 1999.
16. J. Kurose and R. Simha, "A Microeconomic Approach to Optimal Resource Allocation in Distributed Computer Systems," *IEEE Transactions on Computers*, vol. 38, no. 5, pp. 705–717, 1989.
17. C. Langton, R. Burkhart, M. Daniels, and A. Lancaster, "The Swarm Simulation System," http://www.santafe.edu/projects/swarm, 1999.
18. R. T. Maheswaran, Çagri Imer, and T. Başar, "Agent Mobility Under Price Incentives," in *Proceedings of 38th IEEE Conference on Decision and Control*, Phoenix, AZ, pp. 4020–4025, 1999.
19. A. Mohindra, A. Purakayastha, and P. Thati, "Exploiting Non-Determinism for Reliability of Mobile Agent Systems," in *Proceedings of the International Conference on Dependable Systems and Networks*, New York, NY, pp. 144–156, 2000.
20. T. Muldner, "Mobile Computing at Acadia University," Dartmouth College Computer Science Colloquium Slides at http://evilqueen.acadiau.ca/presentations/mobileagents.ppt., 1998.
21. Object Management Group, "Mobile Agent System Interoperability Facility," ftp://ftp.omg.org/-pub/docs/orbos/98-03-09.pdf., 1998.
22. T. Poutanene, H. Hinton, and M. Stumm, "NetCents: A Lightweight Protocol for Secure Micropayments," in *USENIX Workshop on Electronic Commerce*, USENIX Association, pp. 25–36, 1998.
23. O. Regev and N. Nisan, "The POPCORN Market—An Online Market for Computational Resources," in *Proceedings of the First International Conference on Information and Computation Economies*, ACM Press: Charleston, SC, pp. 148–157, 1998.

24. O. Shehory, K. Sycara, P. Chalasani, and S. Jha, "Agent Cloning: An Approach to Agent Mobility and Resource Allocation," *IEEE Communications*, vol. 36, no. 7, pp. 58–67, 1998.
25. I. E. Sutherland, "A Futures Market in Computer Time," *Communications of the ACM*, vol. 11, no. 6, pp. 449–451, 1968.
26. C. F. Tschudin, "Open Resource Allocation for Mobile Code," in *Proceedings of The First Workshop on Mobile Agents*, Berlin, pp. 186–197, 1997.
27. C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta, "Spawn: A Distributed Computational Economy," *IEEE Transactions on Software Engineering*, vol. 18, no. 2, pp. 103–117, 1992.
28. J. E. White, "Telescript Technology: Mobile Agents," General Magic White Paper, 1996.