

Computational Markets to Regulate Mobile-Agent Systems

Jonathan Bredin[†], Rajiv T. Maheswaran[‡], Çagri Imer[‡],
Tamer Başar[‡], David Kotz[†], and Daniela Rus[†]

October 30, 2000

Abstract

Mobile-agent systems allow applications to distribute their resource consumption across the network, but naive shared-resource consumption is not efficient as is evident in the tragedy-of-the-commons phenomenon. 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.

1 Introduction

Breakthroughs in high-speed computing and network bandwidth have surpassed progress to reduce network latency. As computers have become more economical and standardized, computation is no longer the constraining factor in

[†]Department of Computer Science, Dartmouth College, 6211 Sudikoff Lab, Hanover, NH 03755

[‡]Coordinated Science Laboratory, University of Illinois, 1308 W. Main Street, Urbana, IL 61801

distributed-system 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 latency is exaggerated by the greater use of wireless and intermittently connected networks. A technique for reducing an application's end-to-end latency is to move computation closer to scarcer resources, avoiding latency incurred in network communication.

A mobile agent is a user program with the ability to autonomously move from one host and resume execution at another, and a mobile-agent system provides the mechanisms for decentralizing resource allocation by relocating computation represented by mobile agents. 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 and that the quality of resources allocated to them should reflect their priority.

Mobile-agent systems provide a decentralized flexible architecture on which to base network management [3] and handle user preferences [15]. Mobile agents have been used to resubmit tasks to alternate sites by applying knowledge of the loads on various resources [19, 23]. Other applications for mobile agents are to enhance video conferencing performance [2]; to allow users to 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 [13] lead us to believe that a standard for mobile agents will emerge, facilitating their use on almost any type of network. The challenge that we face is to find a model that captures the priority of the various tasks in a network, and to determine an allocation policy where priorities are respected and where agents need only know their own preferences regarding the completion of their tasks.

Ideas from economics provide solutions to many of the issues discussed above. Market-based resource allocation has been used in various distributed resource-allocation problems including computing resources, network bandwidth, and manufacturing systems [9]. For example, Gagliano, et al., applied auction mechanisms to allocate computer resources [11], and Kurose and Simha studied microeconomic approaches for distributed resource allocation [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 [26, 7].

Computational markets rely on some form of currency exchange. At the most basic form, an agent's currency represents its potential to act in the network. At a higher level, the currency might represent actual legal tender necessitating security and cryptographically verifiable electronic cash [12, 21].

In this paper, however, we address a different type of economic market where the seller's role is subdued and the competition for the resources is solely between the agents. This situation occurs in semi-closed systems such as military networks, corporate intranets, or open systems where the users have already passed a higher-level admission procedure and the servers are non-idling. The objective is for agents to complete their tasks as fast as possible while still al-

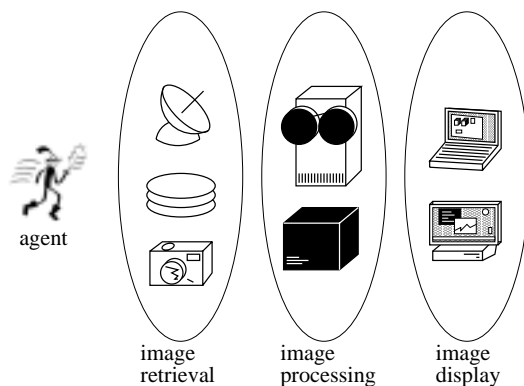


Figure 1: An example mobile-agent itinerary. The agent must visit one host from each group and choose at what priority to execute at each visited host.

lowing users to prioritize their agents. Implicit in this goal is that an agent has no preference for its remaining endowment at the end of its lifetime. The total processing time for any set of tasks is minimized when the utilization factor is maximized, i.e., when no system resources are idle. When multiple agents simultaneously require the use of a resource at a node, however, a bargaining mechanism is necessary to decide which agents will first receive access to that resource and how much of that resource each will receive. The resource itself has no preference about how it is partitioned among the agents as long as it is being fully utilized.

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.

When several agents need access to a particular CPU at the same time,

that CPU schedules the different tasks based on their priorities. The CPU will operate at its maximum rate regardless of the number of tasks requiring service or the allocation scheme.

In this paper we propose an allocation scheme where agents compete for resources. We assume each agent is endowed with electronic cash from its user. The amount of the endowment is the result of a higher level optimization problem that we do not address in this paper. The agent is allowed to spend as much of this endowment as necessary to purchase resources to complete its tasks as quickly as possible.

We derive an agent's optimal response function given the agent's endowment and the state of the network. We further show that there is a unique Nash equilibrium solution among agents competing for a resource when they bid using the optimal response functions. The total cash being spent at a particular resource serves as an indicator for congestion or demand for that resource, which is the feedback information used by agents to determine their bids.

Under our prioritization policy, our simulations show that agents from most endowment levels achieve faster mean performance than in a traditional mobile-agent computation allocation policy. We also compare our policy with other policies to show that the cost of prioritizing agents is small. Furthermore, we show simulations that demonstrate that our allocation and planning algorithms are insensitive to errors in job-size estimation by agents and are no more sensitive to network delay than traditional policies.

The paper is organized as follows. In Section 2, we formalize the structure of our economy and present our allocation mechanism. In Section 3 we derive

the optimal response for a single agent in the absence of market response to the agent’s demand. In Section 4, we show how to compute an allocation that satisfies all agents’ response functions for a single resource and that the resulting allocation is a Nash equilibrium. In Section 5, we prove the uniqueness of the equilibrium. In Section 6, we discuss how our allocation mechanism would be implemented in a network. In Section 7, we show simulation results that demonstrate the effectiveness of our algorithm under varying workloads, network delays, and job-size estimation error. In Section 8, we describe some related work. 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 notation used in the paper.

2 System Model

Users assign agents a series of tasks to be completed sequentially. Each task is characterized by q_k^i , which denotes the size of the k -th task in the i -th agent’s itinerary. For the image retrieval example in Section 1, task size could be the expected number of CPU clock cycles required to complete the task. The user wishes the agent to complete its tasks as fast as possible and endows the i -th agent with e^i dollars of electronic currency to compete for network resources. We assume that the user has solved a higher level optimization problem based on knowledge of previous job completion times given various endowments to determine e^i , which will be taken as a given in this paper. If $q^i = \sum_{k=1}^{K^i} q_k^i$, where K^i is the number of tasks in the i -th agent’s itinerary, 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 that user. A larger ρ^i would indicate an ability to purchase larger portions of resources for each unit of job needed to be completed, enabling the agent to finish more quickly. It could indicate that the particular task sequence is more important (emergency messaging versus low-level maintenance) or that the user is of higher priority to the network and has more capital to spend.

The resources needed by the agents are located at various nodes throughout the network. In our analysis, we assume that network delay between servers for sequential tasks is fixed and therefore the portion of the completion time due to these delays cannot be altered by a scheduling mechanism. Let c_k^i denote the capacity of the resource used by the i -th agent's k -th task. With respect to our subnet example, c_k^i could be the clock speed of the CPU at the computer where the k -th computation of the i -th agent is completed. When several agents request access to the CPU, a scheduling mechanism is needed to partition the clock cycles among the agents. We propose the following mechanism for this purpose: If the i -th agent is willing to pay u_k^i dollars per second and θ_k denotes the total amount that all the agents at that resource are willing to pay, the i -th agent will receive service at a rate v_k^i given by:

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

where $\theta_k^{-i} := \theta_k - u_k^i$ represents the sum of payments from all agents except the i -th agent. For example, if a host accommodates three agents bidding one, two, and three dollars per second, respectively, the agents will receive access to one sixth, one third, and one half of the host's rate of computation. This allocation

is fair in the sense that each agent has the same cost per unit of computation.

If the i -th agent received service at rate v_k^i , the time required to execute the k -th task would be:

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

and the cost of completing the k -th task would be:

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

Our objective is to find an allocation policy for a resource given the structure described above. 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. The optimal bid is a function of the payments of the other agents at the resources in the itinerary, $\{\theta_k^i\}_{k=1}^{K^i}$. 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 all 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 i -th 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 i -th agent's problem as:

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

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

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

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:

$$\frac{\partial \mathcal{L}}{\partial u_k^i} = \frac{-q_k^i \theta_k^{-i}}{c_k^i u_k^{i2}} + \lambda \frac{q_k^i}{c_k^i} = 0 \quad \Rightarrow \quad \lambda = \frac{\theta_k^{-i}}{u_k^{i2}}. \quad (6)$$

Since we are dealing with the i -th 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 k -th server an agent 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 :

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

Incorporating the inequality constraint, we get

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

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:

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

Solving the previous equation for u_1 , we get

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

which yields the bid for the first job for the i -th agent as a function of θ_k^{-i} for $k = 1, \dots, K^i$, 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 i -th 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 at other nodes, which serves as an indicator for congestion or demand at the servers in the itinerary of the agent. We streamline Equation 10 by using place-holder variables and reinstall the i superscript for use in future sections:

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

where

$$\alpha^i := e^i - \sum_{k \neq 1} \frac{q_k^i}{c_k^i} \theta_k^{-i}, \quad (12)$$

$$\beta^i := \frac{q_1^i}{c_1^i}, \quad (13)$$

$$\gamma^i := \sum_{k \neq 1} \frac{q_k^i}{c_k^i} \sqrt{\theta_k^{-i}}. \quad (14)$$

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 and we require that the bids be non-negative. 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 $f_i(\theta_1^{-i})$ will return a positive value if and only if there is a feasible solution. To capture the possibility that the i -th agent will choose not to bid, under certain network conditions (which corresponds to $f_i(\theta_1^{-i}) \leq 0$), we express the bidding strategy as follows:

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

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 the competing 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:

$$\{u_1^i = \max\{0, f_i(\theta_1^{-i})\}\}_{i=1}^N . \quad (16)$$

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 :

$$\{u_1^i = \max\{0, f_i(\theta_1 - u_1^i)\}\}_{i=1}^N . \quad (17)$$

From Equation 17, we obtain an explicit function $g_i(\theta_1) : \theta_1 \rightarrow u_1^i$. Figure 2 illustrates how $f_i(\theta_1 - u_1^i)$ shifts as θ_1 varies. Outside the range $\theta_1 \in (0, \alpha^i/\beta^i)$, $g_i(\theta_1)$ takes the value of 0. We now derive $g_i(\theta_1)$.

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:

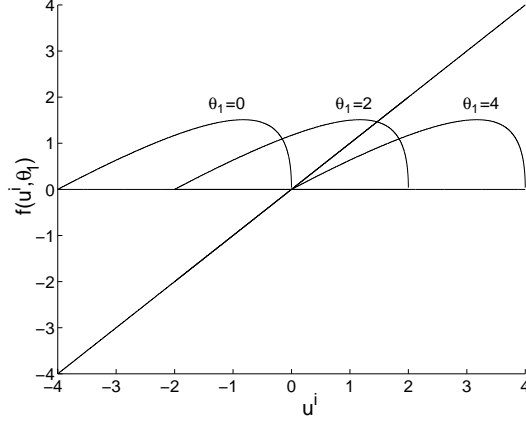


Figure 2: Behavior of $\max(0, f_i(\theta_1 - u_1^i))$ for $\alpha^i/\beta^i = 4$.

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

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

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

Taking the positive root of the equation with respect to u_1 , we have $u_1 = g(\theta_1)$ where

$$g(\theta_1) = \frac{(\alpha - \beta\theta_1)^2}{2\gamma^2} \left(-1 + \sqrt{1 + \frac{4\gamma^2\theta_1}{(\alpha - \beta\theta_1)^2}} \right) \quad (20)$$

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

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

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

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

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

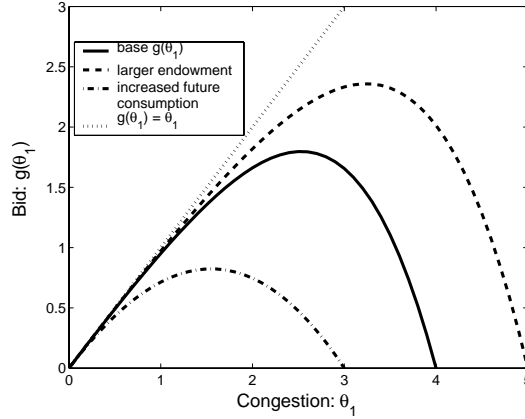


Figure 3: Form of $g_i(\theta_1)$.

Figure 3 displays the shape of $g_i(\theta_1)$ and how it varies. We show an agent's bidding function, $g(\theta)$, and that increasing the agent's endowment allows the agent to submit a larger positive bid over a larger domain compared with the agent's bid given the same level of total demand. An agent with a future load greater than the base agent, but with the same endowment, participates in a smaller set of congestion levels and makes lower bids than the base agent when it does participate, as it must allocate more of its capital to latter jobs.

Returning to our goal to choose the equilibrium bids 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 $\sum_{i=1}^N u_1^i - \theta_1 = \sum_{i=1}^N g_i(\theta_1) - \theta_1 =: h_1(\theta_1) = 0$. 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

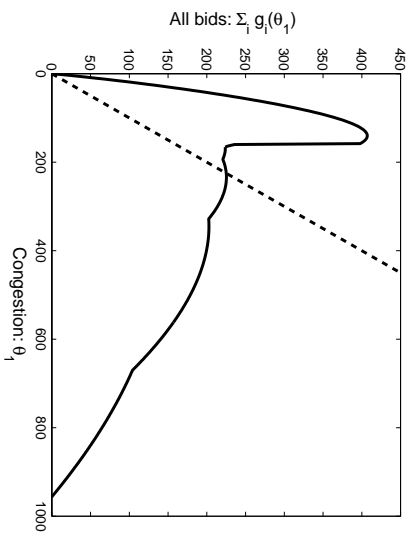


Figure 4: Sample plot of $\sum_{i=1}^N g_i(\theta_1)$ versus θ_1 for 16 agents. Equilibrium occurs at $\sum_{i=1}^N g_i(\theta_1) = \theta_1$ (i.e., $h(\theta_1) = 0$), which we show at the intersection of the dotted line and the plotted curve.

and $h_1(0^+) > 0$. We also know that for the non-trivial 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 $\sum_{i=1}^N g_i(\theta_1)$ versus θ_1 in Figure 4.

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, $g_i(\theta_1)$ is discontinuous. By applying L'Hôpital's rule to Equation 20, however, we see that

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

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 $g_i(\theta_1)$ to yield a solution. Since $g_i(\theta_1)$ 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 $O_1 \supset O_2 \supset \dots \supset O_N$ (i.e., $\alpha^1/\beta^1 > \alpha^2/\beta^2 > \dots > \alpha^N/\beta^N$) 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 $O = \cup_{i=1}^N O_i = (0, \max_i \{\alpha^i/\beta^i\}) = O_1$. We now further strengthen that result. Let us define $h_1^n(\theta_1)$ as follows:

$$h_1^n(\theta_1) = \sum_{i=1}^n g_i(\theta_1) - \theta_1. \quad (23)$$

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

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

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

$$g_i(0) = 0 \forall i \Rightarrow h_1^n(0) = 0. \quad (26)$$

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

Lemma 1 *If $h(x)$ 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.,

$$ah(x) + (1 - a)h(y) < h(ax + (1 - a)y) \quad (27)$$

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 get 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:

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

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 users bidding functions are in the form of Equation 20, but we note that the results hold for any $g(\theta)$ where $g(\theta) > 0$ on some set $(0, b)$, and is zero elsewhere, and further, $g(\theta)$ is concave and twice differentiable on $[0, b]$.

6 Implementation in a Network

Resource allocation in our model is computed on a resource-by-resource basis. Every time an agent arrives at or finishes with a resource, each agent that requires access submits a bidding function characterized by the real-valued positive triple, (α, β, γ) . These values depend on the agent's beliefs about future demand for resources on its itineraries. These estimates allow an agent to budget expenditures for the current task. One method to forecast demand for a resource is to average the total cash spent at that resource over time.

Once agents submit their bidding functions, the node partitions the resource based on the Nash equilibrium allocation. Thus, each agent will be satisfied with the amount and cost of the resource that it receives at the current node, 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 state of that node changes, i.e. another agent arrives or leaves the resource. 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 that an agent “cheats” by submitting a bidding function that is not based on its true beliefs. An agent might hope that an altered equilibrium might facilitate faster itinerary completion. Further optimization of an agent’s bid function proves difficult, however. The optimization requires knowledge of other agents’ private information, reasoning about the possibility of other agents making similar calculations, and access to additional computation. To construct a better bidding function, an agent must reason about the response of its competition’s bidding functions. The calculation of a bid function is based upon private information that is not available to other agents in our model. Further difficulties arise when one considers that all agents may reason about their competition’s bidding functions. One agent taking action affects another’s actions, which in turn affects his. This cascading effect could incur complex and costly analysis, which is discouraged in our model by the facts that each agent pays for its computation with its endowment and that each agent must submit its bidding parameters upon arrival at a resource. The effects of indirect interaction among agents on their beliefs and bidding behavior is an interesting topic for future research.

7 Simulation

In this section, we present results of simulating our resource-allocation policy presented in Section 4. We show that agent performance is highly correlated with endowment; that when our system is overloaded, poorer agents are dropped to maintain higher priority agents' performance; our algorithm is robust to an agent's errors in job-size estimation; and that there is a structure to empirical 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 exponentially distributed number of jobs to complete. Our simulations 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 an agent's endowment size relative to the sum of its job sizes as a truncated Gaussian random variable with a positive mean. 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 offers was picked uniformly at simulation initialization. Host capacity was determined by a positively 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, hosts publish their capacity and

immediate level of congestion, θ , to all agents.

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 time 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 , for hosts offering the corresponding service. Each agent chose the next site to be the one that minimized the sum of network transfer time for the next hop and execution times, 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, $g(\theta)$, and uses a bisection search to

Algorithm 1 Choose Next Site for Agent i

```
1:  $t_{min} := \infty$ ;  $nextHost := \emptyset$ 
2: for all hosts  $k$  offering service next in itinerary do
3:    $t_k := [\text{transferLatency to: } k \text{ from: } currentHost]$ 
       $+ q_k g(\theta_k^{-i}) / (c_k * (\theta_k^{-i} + g(\theta_k^{-i})))$ 
4:   if  $t_{min} > t_k$  then
5:      $t_{min} := t_k$ ;  $nextHost := k$ 
6:   end if
7: end for
8: return  $nextHost$ 
```

find the bidding level where $\theta = g(\theta)$. This search is conducted every time an agent arrives or departs the host. Algorithm 2 sketches the operation.

Algorithm 2 Allocate Resources for Host k

```
1: while true do
2:    $t :=$  time since last arrival/departure
3:   for all agents  $i$  do
4:     deduct  $t g_i(\theta)$  from agent  $i$ 's endowment
5:   end for
6:   add new agent or remove departing agent
7:   for all agents  $i$  do
8:     query agent  $i$  for  $\alpha, \beta$ , and  $\gamma$  (Equations 12-14) to build  $g_i(\theta)$ 
9:   end for
10:  search for  $\theta = \sum_{i=1}^N g_i(\theta)$  in  $(0, \max_i(\alpha_i/\beta_i))$ 
11:  for all agents  $i$  do
12:     $v_k^i := c_k g_i(\theta) / \theta$ 
13:  end for
14: end while
```

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 time and the execution time of the 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 that they currently host.

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 the time an agent would take to complete a job at the second host without any competition, q_k/c_k .

We ran three experiments. First we verified that our game-theoretic resource-allocation policy prioritizes agents by endowment. We compared the perfor-

mance 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’ endowments with their performance. After the network reached a steady state, we designated seven percent 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 endowment sizes, spanning two standard deviations, σ , around the mean endowment, μ , and differing task sizes.

Figure 5 shows how endowment affects agent performance in one experiment and how agents performed in separate simulations using the shared, FCFS, and SRPT policies. The data labeled “GT” shows the performance of agents operating under the game-theoretic allocation policy. We plot the mean and standard deviations of agents’ performance versus their endowment level. Endowment is an agent’s job size sum divided by the amount of currency with which the agent begins. We also plot the mean performance of all agents operating under the shared, SRPT, and FCFS allocation policies in separate simulations. In the

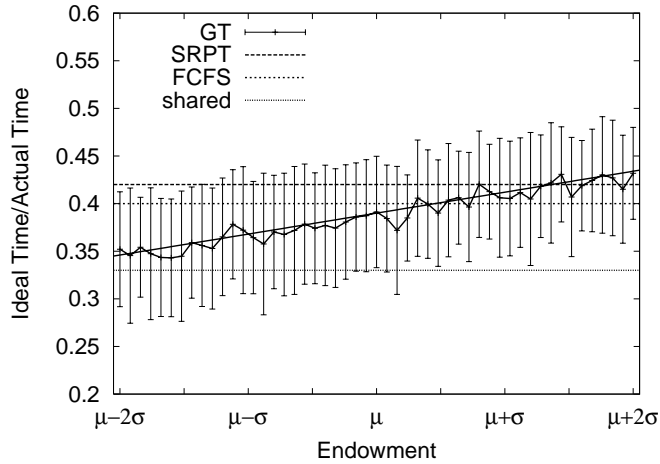


Figure 5: Endowment versus ideal time relative to actual time. For the game-theoretic (GT) approach, we plot the observed means in heavy block lines, standard deviations with error bars for agents at each endowment level for two standard deviations, σ , of endowment around the mean, μ , and a linear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

experiment, agents' job sizes had 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 non-congested itinerary completion time and the actual itinerary completion time for agents. A higher number indicates better performance. In the experiment, agents with higher endowments tend to perform better on the average than those with lower endowments. The mean performance of agents using the shared policy was 0.33, while agents across all endowment levels using the game-theoretic policy achieved a mean performance of 0.39 – an improvement of 18 percent. 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 eight percent better than mean performance

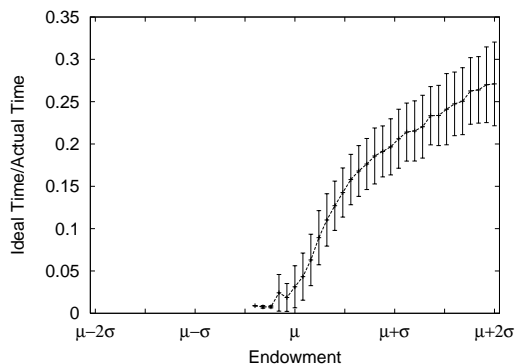


Figure 6: Endowment versus ideal time relative to actual time when agent requests exceeded capacity. We plot observations two standard deviations, σ , around the mean endowment μ .

of agents across all endowment levels operating under the game-theoretic policy, however, neither policy prioritizes agents.

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 endowments 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 endowment affected agent performance when agent re-

quests exceeded system capacity. We observed that agents with endowments less than the mean were not able to complete their itineraries at all. Among agents with larger than average endowments, there was a strong correlation of endowment 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

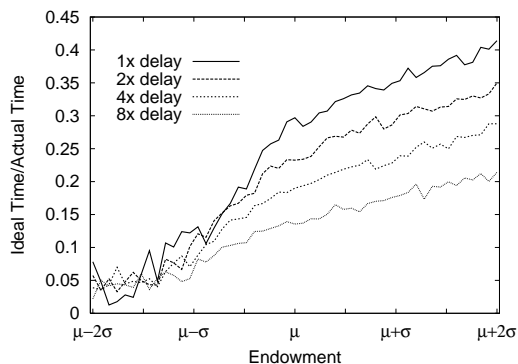


Figure 7: We plot the effectiveness of prioritization of the allocation policy at different network delays and agent endowment levels. We plot observations two standard deviations, σ , around the mean endowment, μ .

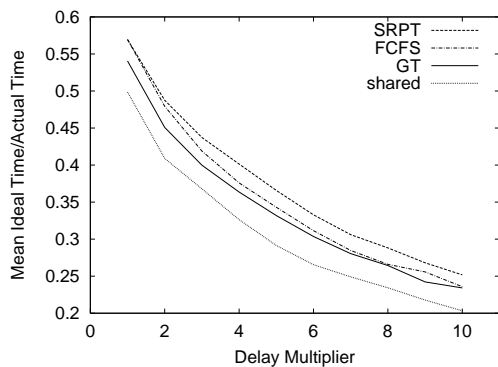


Figure 8: The mean performance of agents operating in various allocation policies versus network delay.

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 aged, ideal/actual performance ratio gradually decayed and that 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

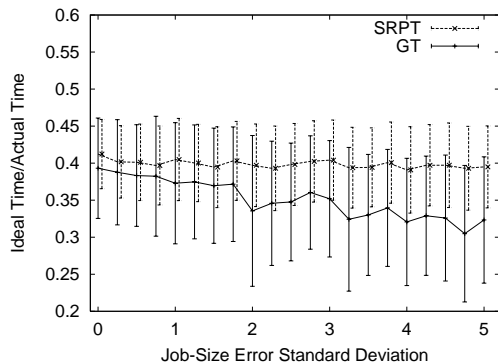


Figure 9: Agent performance versus the standard deviation of job-size error.

endowment levels operating 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 agents' performance in simulations that varied all agents' 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 increases. Agents experienced approximately a three percent decrease in performance for every multiple increase 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 in Figures 10 and 11 for experiments that used exponentially and Pareto distributed job sizes, respectively, to see if there were any structure in bid fluctuation. 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 than 0.03. The experiment using 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

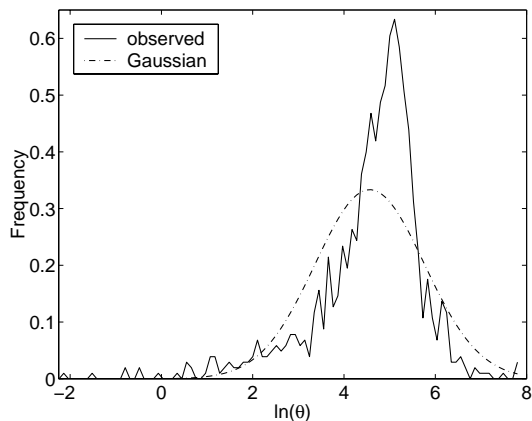


Figure 10: A histogram of bid sum on a log scale, $\ln(\theta)$, observed at a server where agents arrived with exponentially distributed job sizes, and the corresponding Gaussian distribution.

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.

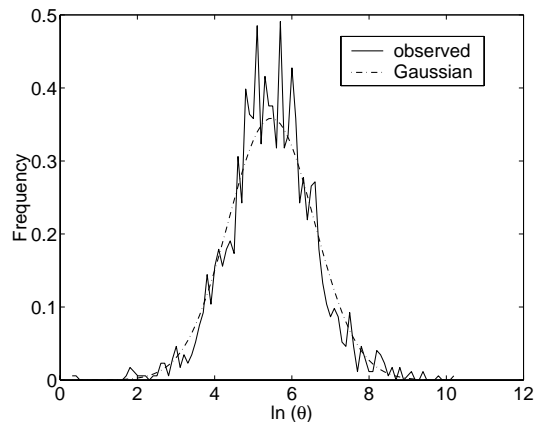


Figure 11: A histogram of bid sum on a log scale, $\ln(\theta)$, observed at a server where agents arrived with Pareto distributed job sizes, and the corresponding Gaussian distribution.

We are not the only group to promote the use of markets in mobile-agent systems. The Geneva Messengers project [25] 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 [27] 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 resource. A more general policy would be for hosts to issue a common permit in the form of a verifiable electronic currency.

POPCORN [22] 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 paral-

lel 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 [24]. Spawn is perhaps the most cited work dealing with computational economic systems [26]. 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. Clearwater et al. use double auctions 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 ten percent.

Boutilier et al. take an approach similar to ours in solving sequential resource-allocation problems [4]. They overcome incentive compatibility problems with buyers submitting their true evaluations by having each buyer express her preference to an agent identical to her competitors. 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 another centralized 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 where participants have convex utility functions and there is gross substitutability among goods (goods are not complementarities for one another), but often converges to a solution even when gross substitutability is violated.

9 Conclusion

We describe in this paper 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 at the node. 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 optimal policy to minimize 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 percent 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 is an open challenge.

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.

Acknowledgements

This work was supported in part by Department of Defense contract MURI F49620-97-1-0382, DARPA contract F30602-98-2-0107, and an EPRI/ARO contract.

A Proof of Concavity of $g_i(\theta_1)$

Theorem 2 *An agent's bidding function, $g_i(\theta_1)$, 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:

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

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

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

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

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

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

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

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

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

$$v(b, x) = \frac{-4\alpha b^2 x^3 + 12\alpha b x^4 + 3b^2 x^4 + 8x^6}{-12bx^5 - 8(x^4 - bx^3 + b\alpha x^2)^{\frac{3}{2}}} \quad (33)$$

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

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

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 . \diamond

B Notation

α^i	$:= e - \sum_{k \neq 1} \frac{q_k^i}{c_k^i} \theta_k^{-i}$ 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	rate at which the k -th host that the i -th agent visits can compute
e^i	i -th agent's endowment.
$f_i(\theta_j^{-i})$	i -th agent's optimal bid conditioned on all other agents' bids at the j -th host sum to θ_j^{-i} , and $\theta_j^{-1} \in (0, \alpha^i/\beta^i)$.
$g_i(\theta_j)$	i -th agent's bid conditioned on all agents' bids at the j -th host sum to θ_j , and $\theta_j \in (0, \alpha^i/\beta^i)$.
$h_i^k(\theta)$	the difference of θ and the sum of $g_i(\theta)$, where the sum is over the k agents with the largest positive bidding domains.
K^i	number of tasks in the i -th agent's itinerary.
m_k^i	i -th agent's expenditure at the k -th host that it visits.
q_k^i	size of the i -th agent's k -th job.
t_k^i	time taken to complete the i -th agent's k -th task.
θ_k^{-i}	sum of the bids of all other agents visiting the k -th host that the i -th agent visits.
θ_k	sum of the bids of all agents visiting the k -th host.
u_j^i	amount that the i -th agent bids at the j -th site it visits.
v_k^i	rate at which the i -th agent computes its k -th job.

References

- [1] T. Başar and G. J. Olsder. *Dynamic Noncooperative Game Theory*. Classics in Applied Mathematics, SIAM, 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*, pages 273–284, Stuttgart, Germany, Sept. 1998.
- [3] A. Bieszczad, B. Paturek, and T. White. Mobile agents for network management. *IEEE Communications Surveys*, Sept. 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, 1999.
- [5] J. Bredin, D. Kotz, and D. Rus. Utility driven mobile-agent scheduling. Technical Report PCS-TR98-331, Dartmouth College, May 1998.
- [6] K. Calvert and E. Zegura. GT-ITM: Georgia Tech internetwork topology models, 1996. <http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm/gt-itm.tar.gz>.
- [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*, Marina Del Ray, CA, 1997. ACM Press.
- [8] J. Q. Cheng and M. P. Wellman. The WALRAS algorithm: A convergent distributed implementation of general equilibrium outcomes. *Journal of Computational Economics*, 12:1–23, 1998.

- [9] S. H. Clearwater, editor. *Market-Based Control*. World Scientific, Singapore, 1996.
- [10] S. H. Clearwater, R. Costanza, M. Dixon, and B. Schroeder. Saving energy using market-based control. In Clearwater [9], pages 253–273.
- [11] R. Gagliano, M. Fraser, and M. Shaefer. Auction allocation of computer resources. *Communications of the ACM*, 38(6):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*, 1(1), Winter 1996.
- [13] T. O. M. Group. Mobile Agent System Interoperability Facility, 1998. <ftp://ftp.omg.org/pub/docs/orbos/98-03-09.pdf>.
- [14] D. Johansen. Mobile agent applicability. In *Proceedings of the Second International Workshop, Mobile Agents '98*, pages 80–98, Stuttgart, Germany, 1998.
- [15] D. Kotz and R. S. Gray. Mobile agents and the future of the Internet. *ACM Operating Systems Review*, 33(3):7–13, Aug. 1999.
- [16] J. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Transactions on Computers*, 38(5):705–717, May 1989.
- [17] C. Langton, R. Burkhart, M. Daniels, and A. Lancaster. The Swarm simulation system, 1999. <http://www.santafe.edu/projects/swarm>.
- [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, Dec. 1999.
- [19] A. Mohindra, A. Purakayashita, and P. Thati. Exploiting non-determinism for reliability of mobile agent systems. Technical Report RC21591, IBM Research, 1999.
- [20] T. Muldner. Mobile computing at Acadia University. Dartmouth College Computer Science Colloquium, 1998. Slides at <http://evilqueen.acadiau.ca/presentations/mobileagents.ppt>.
- [21] T. Poutanene, H. Hinton, and M. Stumm. NetCents: A lightweight protocol for secure micropayments. In *USENIX Workshop on Electronic Commerce*, pages 25–36. USENIX Association, September 1998.
- [22] 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 Economics*, pages 148–157, Charleston, SC, Oct. 1998. ACM Press.
- [23] O. Shehory, K. Sycara, P. Chalasani, and S. Jha. Agent cloning: An approach to agent mobility and resource allocation. *IEEE Communications*, 36(7):58–67, July 1998.
- [24] I. E. Sutherland. A futures market in computer time. *Communications of the ACM*, 11(6):449–451, June 1968.
- [25] C. F. Tschudin. Open resource allocation for mobile code. In *Proceedings of The First Workshop on Mobile Agents*, pages 186–197, Berlin, April 1997.
- [26] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering*, 18(2):103–117, Feb. 1992.
- [27] J. E. White. Telescript technology: Mobile agents. General Magic White Paper, 1996.