Utility Driven Mobile-Agent Scheduling
Dartmouth Technical Report PCS-TR98-331*

Jonathan Bredin, David Kotz, and Daniela Rus
Department of Computer Science
Dartmouth College
Hanover, NH 03755
{jonathan, dfk, rus}@cs.dartmouth.edu

October 3, 1998

Abstract

Mobile agents are programs capable of migrating from one host machine to another. We propose that mobile agents purchase resource access rights from host machines thereby establishing a market for computational resources and giving agents a metric to evenly distribute themselves throughout the network. Market participation requires quantitative information about resource consumption to define demand and calculate utility.

We create a formal utility model to derive user-demand functions, allowing agents to efficiently plan expenditure and deal with price fluctuations. By quantifying demand and utility, resource owners can precisely set a value for a good. We simulate our model in a mobile agent scheduling environment and show how mobile agents may use server prices to distribute themselves evenly throughout a network.

1 Introduction

Mobile agents are programs that may, under their own volition, jump from one host and resume execution at another host. We propose using electronic

*Revised October 3, 1998

markets to regulate agent abilities and to create incentive for host owners to allow foreign agents access to their domains. This method of regulation requires hosts (resource owners) setting prices for outside access and allows incoming agents (resource buyers) to buy access with an abstract currency.

We wish to use a market system to regulate agents, encourage them to act responsibly, and evenly distribute themselves throughout the network. Ideally, prices and congestion will be highly correlated: a congested resource will become more expensive. Participants have a finite amount of currency, thus agents have incentive to attempt to evenly spread themselves throughout the network. This has the effect of implicitly distributing the decision-making processes avoiding a central point of control. Finally, a large group of independent agents can be thought of as a community, which gives an experimental platform for developing and testing coordination algorithms inspired be human societies.

For example, a mobile agent may migrate to a machine to access a particular database. The agent arrives with some amount of electronic cash accepted by the server. The agent may plan to save some of the cash for future points in its itinerary or simply return the change to the user. The server's owner then uses the proceeds of the sale to endow her own agents or possibly transfer the revenue into some other currency, say U.S. dollars.

The agent might express a preference to pay higher prices for faster service. The server then calculates a price and a resource allocation for all present agents.

As more agents arrive at the site, the server will most likely raise the price of service to maximize profits. Rising prices encourage prospective agents to look elsewhere. This gives agents a mechanism to compare server congestion as well as incentive to distribute themselves evenly across the network.

There are several methods of pricing electronic markets proposed in [GSW97a, MMMM95, WHH$^+$92, WWWMM98, CW98], but little work has been done on calculating mobile agents' or users' demand and utility. In this context, utility denotes the satisfaction that a user or agent derives from consuming a given resource. A commonly used utility model in microeconomics is the Cobb-Douglas utility function [PR92], which we apply to generate functions to model user demand and simulate agent resource markets.

In this model, an agent's *demand function* takes the market price for a good and determines the amount that the agent would like to buy. A demand function could be used by agents to determine when and how much of a resource to buy. Individual agents' demand functions are derived from utility functions.

A market-demand function determines how much the market, as an aggregate entity, will buy at any given price. To set prices, sellers must estimate the market-demand function. In doing this, resource owners discover information necessary to optimally price their resources.

Normally, a vendor must take its competitors' immediate actions into account. In our model, competing vendors are typically located at other sites and agents may only deal with those vendors currently at the site. However, agents have the capability to relocate to more a desirable location to deal with other vendors and we assume that agents have some knowledge about prices of resources throughout the network.

Market-demand functions are calculated from the demand of individuals participating in the market. A natural problem that arises is how the seller acquires information concerning individual demand. In this paper we examine a solution that uses a form of auction where buyers submit demand functions, which sellers use to compute competitive prices. An important aspect of this approach is setting up the auction so that there is incentive for prospective buyers to be honest in their bidding.

We study this style of market with the intent of applying it to an existing mobile-agent system, D'Agents [Gra97], and to other related applications. One problem mobile-agent system designers face is how to allocate resources to agents. Here, we define resources as access to any computing device or service an agent may use. Examples include compute or network access time and database access. We hope to find empirical evidence that markets provide agents with the motivation to evenly distribute themselves throughout the network.

In this paper we present a formal utility model and evaluate its use in a mobile-agent scheduling system. In particular, we examine its effectiveness in balancing load and the overhead of information required to drive the system.

This paper is organized as follows. We first discuss why mobile agents require regulation and why markets are a good candidate method. We briefly cover existing relevant work. Then we present a utility model and show how agent servers can compute efficient prices for a scheduling resource. We discuss a simulated implementation of our model and we conclude with some remarks on our future work that concerns load balancing and mitigating mobile agents' risk through the use of call options.

2 Motivation

Consider a mobile-agent system in which agents migrate to a random host to perform their computation. Initially, for simplicity, assume that all hosts are identical. An active unregulated mobile-agent system has a propensity to allow significant groups of agents to accumulate at a single site. A quick calculation gives us a rough estimate of the expected highest load in the network by using a standard application of the Chernoff bound [SW94]. If X is a sum of independent random variables, X_i, where each $X_i \in [0, 1]$, then for any $\epsilon > 0$:

$$\Pr[X \leq (1 + \epsilon)\mu] \leq e^{-\mu \min(\epsilon, \epsilon^2)/3}$$

where μ is the expected value of X. Let n denote the number of agents in the system and X_i the event that the i^{th} agent jumps to a given host. Assuming that the Chernoff bound is tight and that agents have an equal chance of ending up at any one of m hosts, then we would expect some site to have a reasonable chance of hosting a cluster of agents of size:

$$n/m + \max(\sqrt{3c \ln m}, 3 \ln m)$$

Where n/m is the expected number of agents per site. The estimate is obtained by setting the right hand side of the Chernoff bound to $1/m$ and solving for ϵ. If there is a $1/m$ chance that more than this number of agents will jump to a given site, then there is approximately a $1-(1-1/m)^m$ chance of at least one site experiencing this load.

2.1 Markets

Since hosts risk having a significant additional load and expose themselves to possible security problems by allowing additional users, there is little incentive for a system owner to open up their system to a mobile-agent system, greatly limiting mobile-agent applications.

A network such as a mobile-agent system has a value that is quadratic to the number of users. By creating incentive for more hosts to participate in the mobile-agent system, we increase its intrinsic value.

We propose to establish a system that rewards owners for opening their resources to the public: allow resource owners to sell access to outside users and have resource owners distribute the proceeds to users at their sites, who will in turn give cash to their own agents to use elsewhere. Essentially, this creates an economy of agents.

In addition to reimbursing resource owners, the use of a market to control mobile agents has the potential to evenly distribute agent load throughout the network. Generally, there is a strong correlation between consumer demand (i.e. generating congestion) and higher prices. Thus a site experiencing a high load should set a higher price to maximize profits and to encourage agents to move to a site charging a lower price.

2.2 Utility

A market's efficiency depends on the consumers' ability to assess their needs and then make rational decisions that maximize their utility. Here, utility is a measure of the pleasure a market participant derives from consumption of a good. There are few studies about the source or measurement of agent utility in an electronic market.

Without formalizing utility, there is little certainty in the validity of economic choices. Here we apply well-known utility models to market-structured agent systems and examine the results. Primarily, we are concerned with three attributes that affect agents.

The first property is cost. We limit every agent's monetary resources, so an agent that pays a higher price for service effectively limits the amount of utility it can generate in the future. This is an example of how currency can be considered an abstract good representing future consumption, even though cash holdings have no direct immediate value.

The second property we use is quality of service measured in terms of completion time. This is dependent on resource congestion and the hardware providing service.

There could be other qualities besides completion time. In information-retrieval tasks, for example, accuracy is an issue. This can be a difficult concept to measure algorithmically, so we will consider it in this paper.

Finally, there may be some question of an agent's chance of successfully completing a task. For some users, there is utility in risk avoidance. It would be perfectly reasonable for users to negotiate the level of risk based upon their preferences. For example, a user might want to pay more for a task that will complete with high probability. Conversely, a user might expect a discount on a service of questionable reliability.

3 Related Work

There is a substantial body of work on pricing in electronic markets. There are two methods: systems can compute the optimal equilibrium prices exactly or estimate the equilibrium prices based upon the past.

We are primarily interested in the former, and we build upon ideas taken from WALRAS [CW98] and Smart Auctions [MMV95] where sellers compute equilibrium prices by soliciting bids from buyers. In WALRAS, bids are demand functions constructed from a vector of current prices. There is an auctioneer for every good. Bids are submitted to the auctioneers, who compute a price to match supply and demand. The auctioneers notify prospective buyers of the new prices so that they may recalibrate demand functions for future iterations. The process repeats until prices reach a steady state.

Smart Auctions uses a simpler algorithm. Here, sellers attempt to sell some fixed quantity of a good and receive price bids from buyers. The winning bid is the highest bid, but the price is assigned to be the highest losing bid, giving buyers incentive to submit honest bids.

If the process of computing prices exactly is costly or otherwise infeasible, a different approach is to assume that an equilibrium exists and that a mature market has an equilibrium that it is normally near. Using the recent past and feedback from user consumption, it is possible to adjust past prices to compute efficient current prices [GSW97b].

4 A Utility Model

To allow agents to plan their expenditures, we establish a formal goal at which agents aim when making decisions. In economic systems, participants generally try to maximize their utility, but we have yet to quantify utility or see it quantified elsewhere. We use a modification of the traditional Cobb-Douglas model from textbook microeconomics.

The Cobb-Douglas utility function is a frequently used method of expressing an individual's utility [PR92]. For two goods, S and R, the Cobb-Douglas utility function is:

$$U(S, R) = a \ln(S) + (1 - a) \ln(R) \tag{1}$$

where S and R are the quantities consumed and a is some real number in $[0, 1]$ that describes an individual's taste for S relative to R. Representing utility in this manner provides us with diminishing marginal utility: a common theme in life; having a lot of something is nice, but having twice as much is not twice as nice.

Consider a scenario where agents are concerned with two qualities: priority in some scheduling system and the amount of cash that they have after being scheduled. This models a situation where a mobile agent jumps to a machine and requests a block of execution time with some general priority. For simplicity, we bundle together all of an agent's computational requirements.

First, we modify the utility function to account for the size of an agent's task:

$$U(S, R) = a \ln(QS) + (1 - a) \ln(R) \tag{2}$$

where Q is the size of the job in some "units," S is the throughput in job units per second, and R is the extent of the agent's remaining currency supply upon completing the task. The product QS weights the importance of the job with respect to its size. Using the product of Q and S instead of the quotient gives us units expressed in time, but execution time is an economic bad for the agent. The use of seconds as units would require us to modify the Cobb-Douglas further.

Agents have a budget constraint of:

$$I = QP + R \tag{3}$$

where I is the agent's initial endowment, QP is the expenditure (quantity Q times price P), and R is remainder (savings).

It is now possible to derive an individual's demand function by solving for the Marginal Rate of Substitution, *MRS,* of performance, S, for savings, R, which we denote as MRS_{SR}. This is the quotient of the partial derivative of utility with respect to performance, $\partial U/\partial S$, and the partial derivative of utility with respect to savings, $\partial U/\partial R$. Here the MRS_{SR} measures how much savings an agent is willing to give up to achieve one more job unit per second performance. At a competitive equilibrium, MRS_{SR} should be equal to the price per job unit, P.

$$\frac{\partial U/\partial S}{\partial U/\partial R} = \frac{aR}{(1 - a)S} = P \tag{4}$$

We use Equation (4) and the budget constraint, Equation (3), to generate a demand function that returns the scheduling priority (in job units per second) an agent would buy for a given price. Note that this function should be restricted to non-negative values.

$$S = \max\left(\frac{a(I - QP)}{(1-a)P}, 0\right) \tag{5}$$

Market demand is computed by composition: for n agents, sum their individual demand functions:

$$S_{total} = \sum_{\text{all agents } i} \max\left(\frac{a_i(I_i - Q_iP)}{(1-a_i)P}, 0\right) \tag{6}$$

I_i, a_i, and Q_i are constants describing the agents wishing to execute at the host. In the short term, S_{total} is fixed and we assume that the sellers have no ability to create additional resources. Given that agents have the ability to travel to other sites selling the same resource, there is competition among host sites. Sellers can maximize both total utility and the profits by selling all of the available resources to agents.

The clearing price for a system with n agents is computed by solving Equation 6 for P yielding:

$$P = \frac{\sum_i \frac{a_i I_i}{1-a_i}}{S_{total} + \sum_i \frac{a_i Q_i}{1-a_i}} \quad \forall i \in [1 \ldots n] : PQ_i \leq I_i \tag{7}$$

When $PPQ_i > I_i$, the first argument of the max() function in Equation 6 is negative, thus the terms drop out from the sum.

Every time an agent arrives at a host site, the host computes the new market demand, though Equation (7) gives only a static solution where all agents arrive at the same time. By adjusting the utility function in Equation (2) to account for consumption in two sessions, Q_1 and Q_2, with service rates S_1 and S_2, time lengths $T_1 = Q_1/S_2$ and $T_2 = Q_2/S_2$, and per unit prices P_1 and P_2, we can arrive at a general solution. The new utility function becomes:

$$U = a \ln\left((Q_1 + Q_2)\left(\frac{Q_1 + Q_2}{T_1 + Q_2/S_2}\right)\right) + (1-a)\ln(I - Q_1P_1 - Q_2P_2) \tag{8}$$

As in Equation (2), the term inside the first logarithm is derived from the product of the size of the task, $Q_1 + Q_2$, and the average throughput. Q_2/S_2 is substituted for T_2 to eliminate a variable.

By computing the MRS_{SR} in the same fashion as the static solution, we arrive at the same solution if T_1 and Q_1 are zero. Otherwise the agent's demand function for the second period is:

$$S_2 = \frac{-Q_2 + \sqrt{Q_2^2 + \frac{4T_1 a Q_2 (I - Q_1 P_1 - Q_2 P_2)}{(1-a)P_2}}}{2T_1} \qquad \text{if } T_1, Q_1 \neq 0 \tag{9}$$

Again, market demand is computed by summing the individuals' demand functions, though now the clearing price must be found using a numerical root-finding method. Locating the clearing price can be done relatively quickly using one of several algorithms from [PTVF92] since the aggregate demand is well behaved around the market-clearing price.

Since hosts have preferences for resource access similar to agents, the host's consumption can be taken into account by including the host's demand in these calculations, and effectively having the host pay itself for access. Note that the host can pay itself as much as it wants to discourage foreign access or increase the priority of local jobs.

5 A Simulation

We would like to investigate a market's efficiency as a load balancing instrument. We have implemented a simulation of an agent scheduler accepting foreign agents having preferences described by our Cobb-Douglas based utility function, Equation (8). Figure 1 shows a diagram of the simulation execution of each server.

5.1 A Single Server

We first proceed by assuming that there is only a single site agents. The simulation is implemented in C++ with numerical routines from [PTVF92].

In our simulation, the server allocates execution time to incoming agents by searching the aggregate demand function for a market clearing price every time an agent enters or exits the system. Agents arrive at a Poisson arrival rate with exponentially distributed job sizes, Q. These two distributions are

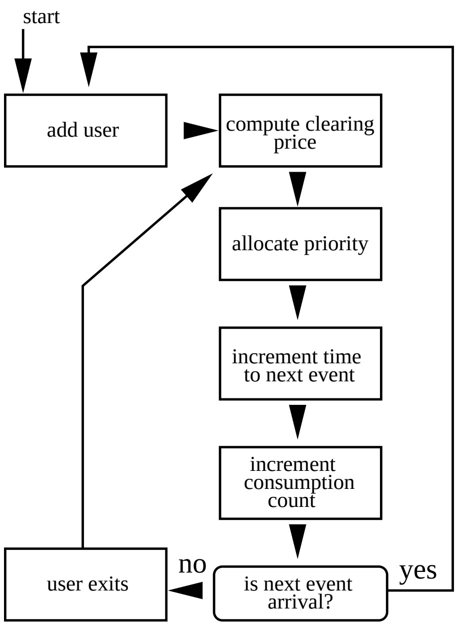

Figure 1: A flowchart illustrating the actions of the simulator.

standard in processor queueing systems. The other two input parameters are agent-scheduling-saving preferences, a, and endowment size, I.

Each agent has an exponentially distributed number of destination hops in its itinerary. The a parameter denotes how much the agent wishes to save for consumption at future destinations and is computed from the ratio of the computation at the next destination to the remaining computation in future hops.

I is computed from a, Q, and a normally distributed random variable defining the agent's owner's view of the job importance to weight the importance of a job linearly with its size. Our decision to use a normally distributed income parameter is arbitrary; the motivation is provide some variation of user preferences.

Users express their preference for scheduling priority by establishing their agent's endowments. The agent then uses the a parameter to express how much of its endowment it would like to spend at a given site. The simulation computes what portion of the original endowment is left given the amount of the job done to adjust I to reflect previous consumption.

The simulation shows what we expected: resource price is positively correlated with the number of users present. To measure market efficiency, we use the Spearman rank correlation coefficient [PTVF92] of price to load, where load is the sum of the sizes of jobs waiting to be completed.

Varying the client arrival intensity has little effect on the load-price correlation. Interestingly, the market requires some variation in user preference to efficiently allocate resources. Figure 2 shows how market efficiency, expresses as load-price correlation, is directly related to the variance in agent endowment. Here we examine the endowment relative to the job size, the per-unit endowment. The variance is expressed as a fraction of the mean.

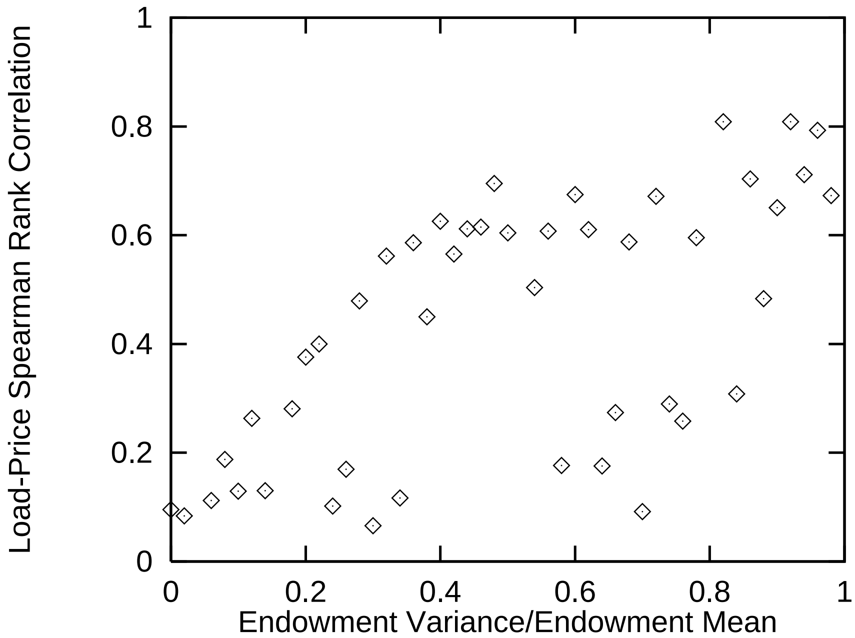

Figure 2: Price-load correlation versus agent endowment variance expressed as the variance of per-unit endowment relative to the mean per-unit endowment.

Given a bursty Poisson arrival process, the market does a surprisingly good job at finding a stable price. Figure 3 shows a histogram of prices through a single simulation. About eighty percent of transactions have a price within five percent of the mean price though there are a few transactions at prices far from the mean. Increasing the load has little effect on the distribution of prices.

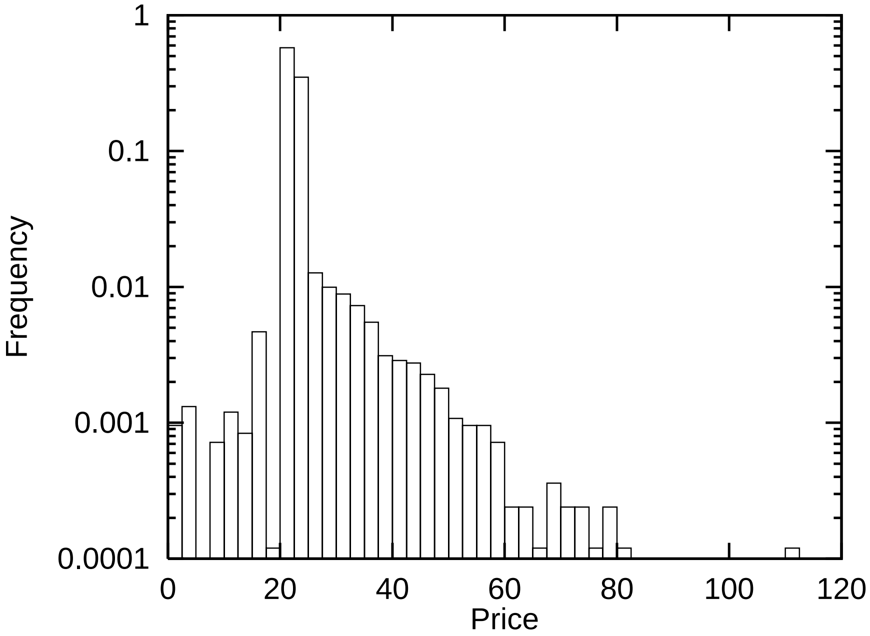

Figure 3: A histogram of the per unit price of computation through a single simulation with twenty percent capacity mean load. Note that the ordinate uses a logarithmic scale.

5.2 Multiple Servers

One of the driving assumptions in the use of markets is that they efficiently and implicitly distribute the decision-making processes. To test this, we constructed a parallel simulation of mobile agents having to choose between multiple identical services at separate sites.

Most load-balancing systems use mean response time as a metric of effectiveness. Since our mobile agents are not only concerned with performance but also with budget expenditure, mean response time is not necessarily a good measure of system performance. Therefore we compare mean client utility in identically seeded experiments to demonstrate the effectiveness of pricing in distributing computing loads.

We make the assumption that clients believe that the additional load incurred by their tasks will be negligible to the server. We leverage this to simplify shopping. The client needs to examine the current price of access at a server and calculate how much the client would like to buy given the price.

As in the previous section, agents have a Poisson arrival rate with exponential distributed job sizes. The variable element is now how agents choose their destination. We examine two possibilities:

1. We may assume that there is perfect price knowledge, i.e, another agent provides accurate up-to-date pricing information, possibly for a fee, though we do not account for any such fees here.

2. An agent might choose only to look at the price of a fixed number of service locations.

For now, we assume that the population of clients use identical strategies.

When we assume that clients' location is fixed at start up, there is the question of how to place the agents. Agents may look at any number of servers, but there might be a cost associated with "shopping." The results of a simulation using ten servers shown in Figure 4 describe the average utility derived when clients randomly choose a server, choose the best among two randomly evaluated servers, choose the best among five randomly selected servers, and choose the best of all ten servers.[1] We use an M/M/10 queue, a single server with the capability of ten, as an ideal baseline comparison, since using an M/M/10 queue is equivalent to assuming that agents can migrate at any time to the "best" host.

The majority of the time, the price at a server occurs near the mean as denoted by the histogram in Figure 3. Because of this, clients generally do not have to check very many servers to find reasonable performance. In our simulation, checking the status of two servers was practically as good as checking all of them.

The clients' ability to find a random server provides a great deal of the load balancing, but the ability to compare the status of a small number of randomly selected servers smoothes the load even more as exhibited by the higher utility values. In fact, checking a small number of servers comes very close to providing ideal load balancing in terms of utility.

6 Discussion and Future Work

We see four important issues in using this model for resource scheduling:

[1] The data has been smoothed with a locally weighted filter to account for noise.

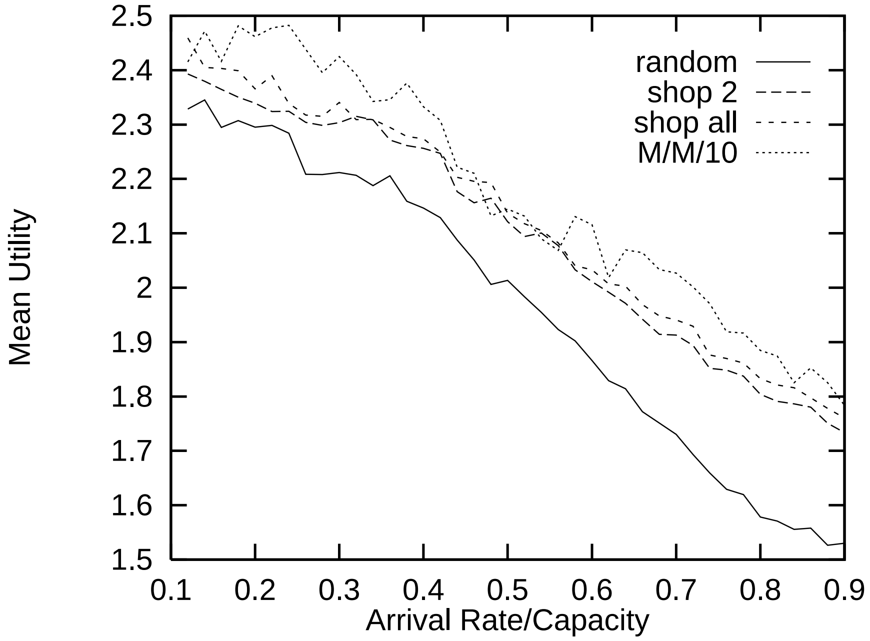

Figure 4: Mean utility versus the arrival rate relative to capacity.

- What are the roles of honesty and trust?
- What issues of policy design need to be corrected?
- How do we make the system even more stable?
- How is it possible to expand the model to handle multiple goods?

6.1 Honesty and Trust

One might argue that agents should not be willing to share budget information with their hosts. If budget information is truly sensitive, the agent should not be trusted with it; however the agent is already at the mercy of the host site, since the host can examine the agent in any fashion it likes [Gra96].

Certainly, hosts that abuse agents will earn a bad reputation, and agents will be reluctant to travel to such sites, thus lowering potential earnings from resource sales. It is possible that important tasks could be performed jointly by several agents in a "chunking" scheme as in [San97] to give hosts incentive to act honestly.

On the other side of the transaction, there is the question of whether it is an agent's best interest to announce its preference for scheduling (a). Intuitively, overvaluing a could oblige an agent to pay an inflated price. On the other hand, undervaluing the parameter could drive the price of scheduling down at the risk of poor performance.

6.2 Cobb-Douglas and Economic Mechanism Design

We do see one drawback with the Cobb-Douglas utility function as applied to this scenario: the utility approaches negative infinity as S and R go to zero, creating an extremely unhappy agent. Initially, this is not a problem if the agent has the ability to assess market conditions and not enter unless there is a reasonably high chance of completion. If market conditions take a downturn midway through an agent's execution, one would expect a utility model to account for the possibility that the agent cuts its losses and exits the market, but it is generally desirable to construct market mechanisms such that an agent will be better off utilizing the mechanism than abstaining from use[WW98].

6.3 Strengthening Stability

One important quality of our model is that it quickly calculates a relatively stable price even given a bursty stream of incoming requests. Approximately 80 percent of all transactions occur within five percent of the mean price.

Infrequently, prices can jump well beyond the mean price. For applications where this is undesirable it would be possible for servers to sell *call options* for scheduling, which are contracts allowing the holder to access the resource for sometime in the future at a prearranged price. Such an option might stabilize prices even further. [SHC96] shows that speculation on these options might further stabilize the market.

Standard stock pricing models are based upon random walks, or modified Weiner processes [Chr97]. In the absence of shifts in demand, our model's prices definitely do not display the behavior of a random walk since prices tend to return to an equilibrium. Further statistical analysis, possibly with a real-life application, will give more insight on how to calculate the value of options to be used in our model.

6.4 Expanding the Market

Finally, our model only allocates one service among agents. In reality, agents consume many resources. Computing general equilibrium prices exactly when the market consists of more than two goods is possibly intractable [Ygg98, PTVF92, p. 379]. We will address the problem of adding more commodities to our market by investigating the needs of actual mobile-agent applications to construct a general profile of consumable resources. This knowledge will allow us to approximate all resources as a single abstract resource representing a weighted bundle of the consumables.

7 Summary

We present a mobile-agent scheduling method based upon first microeconomic principles. Agents are assumed to have preferences corresponding to a Cobb-Douglas utility function, which we use to describe user satisfaction in terms of savings and priority scheduling.

From this utility function, we describe how agents can create a demand function to plan expenditure to maximize their utility. Using agent demand, a host site can calculate optimal scheduling prices and hence an allocation.

We simulate a scheduler using these economic ideas to show that agents can dynamically adapt their consumption habits to account for resource contention. While it is likely that an agent will complete its task, there is uncertainty in performance stemming from price fluctuations. It would be desirable for host sites to sell options to access an agreed upon portion of the resource pool at a fixed price for a period of time. This sort of instrument is essentially an American-style call option.

Acknowledgments

This paper describes research done in the Mobile Agents Laboratory at Dartmouth. This work is supported in part by the Navy and Air Force under contracts ONR N00014-95-1-1204 and MURI F49620-97-1-0382, Rome Labs under contract F30602-98-C-0006, and DARPA under contract F30602-98-2-0107. Robert Gray implemented the core D'Agents system. We are grateful to him for his insights in mobile-agent systems.

References

[Chr97] Neil A. Chriss. *Black-Scholes and Beyond Option Pricing Models*. Mc-Graw-Hill, New York, 1997.

[CW98] John Q. Cheng and Michael P. Wellman. The WALRAS algorithm: A convergent distributed implementation of general equilibrium outcomes. *Journal of Computational Economics*, 1998. To appear.

[Gra96] Robert S. Gray. Agent Tcl: A flexible and secure mobile-agent system. In *Proceedings of the 1996 Tcl/Tk Workshop*, pages 9–23, July 1996.

[Gra97] Robert Gray. *Agent Tcl: A flexible and secure mobile-agent system*. PhD thesis, Dartmouth College, June 1997. Available as Dartmouth Computer Science Technical Report TR98-327.

[GSW97a] Alok Gupta, Dale O. Stahl, and Andrew B. Whinston. Priority pricing of integrated services networks. In McKnight and Bailey [MB97], pages 323–352.

[GSW97b] Alok Gupta, Dale O. Stahl, and Andrew B. Whinston. A stochastic equilibrium model of Internet pricing. *Journal of Economic Dynamics and Control*, 21:697–672, 1997.

[MB97] Lee W. McKnight and Joseph P. Bailey, editors. *Internet Economics*. MIT Press, Cambridge, MA, 1997.

[MMMM95] Jeffrey K. MacKie-Mason, John Murphy, and Liam Murphy. Responsive pricing in the Internet. In McKnight and Bailey [MB97], pages 279–304.

[MMV95] Jeffrey K. MacKie-Mason and Hal R. Varian. Pricing the Internet. In *Public Access to the Internet*, pages 269–314. MIT Press, Cambridge, MA, 1995.

[PR92] Robert S. Pindyck and Daniel L. Rubinfeld. *Microeconomics*. Macmillan Publishing Company, New York, 1992.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, 1992.

[San97] Tuomas Sandholm. Unenforced ecommerce transactions. *IEEE Internet Computing*, 1(6):47–55, November 1997.

[SHC96] Ken Steiglitz, Michael L. Honig, and Leonard M. Cohen. A computational market model based on individual action. In Scott H. Clearwater, editor, *Market-Based Control*, chapter 1, pages 1–27. World Scientific, Singapore, 1996.

[SW94] Henry Stark and John W. Woods. *Probability, Random Processes, and Estimation Theroy for Engineers*. Prentice-Hall, Upper Saddle River, NJ, 1994.

[WHH+92] Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart, and W. Scott Stornetta. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering*, 18(2):103–117, February 1992.

[WW98] Michael P. Wellman and Peter R. Wurman. Market-aware agents for a multiagent world. *Robotics and Autonomous Systems*, 1998. To appear.

[WWWMM98] William E. Walsh, Michael P. Wellman, Peter R. Wurman, and Jeffrey K. MacKie-Mason. Some economics of market-based distributed scheduling. In *Eighteenth International Conference on Distributed Computing Systems*, May 1998.

[Ygg98] Fredrik Ygge. *Market-Oriented Programming and its Application to Power Load Management*. PhD thesis, Lund University, June 1998.