The POPCORN Market – an Online Market for Computational Resources

Ori Regev*
Institute of Computer Science
Hebrew University of Jerusalem
Israel
regev@cs.huji.ac.il

Noam Nisan*
Institute of Computer Science
Hebrew University of Jerusalem,
and Interdisciplinary Center, Herzelia, Israel
noam@cs.huji.ac.il

ABSTRACT

The POPCORN project provides an infrastructure for globally distributed computation over the whole Internet. It provides any programmer connected to the Internet with a single huge virtual parallel computer composed of all processors on the Internet, which care to participate at any given moment. POPCORN provides a market-based mechanism for trade in CPU time to motivate processors to provide their CPU cycles for other peoples' computations. "Selling" CPU time is as easy as visiting a certain web site with a Java-enabled browser. "Buying" CPU time is done by writing a parallel program using the POPCORN paradigm. A third entity in the POPCORN system is a "market" for CPU time, which is where buyers and sellers meet and trade. The POPCORN system may be visited on our web-site: http://www.cs.huji.ac.il/~popcorn.

This paper concentrates on the POPCORN market. The market is a trusted intermediary that is responsible for matching buyers and sellers according to economic criteria.

Our design emphasizes minimal communication requirements and minimal strategic considerations on the part of both buyers and sellers. We implemented several market mechanisms of the single-sided and double-sided auction types. We analyze the economic efficiency of these mechanisms using analytical and simulation methods. Our findings support the use of these mechanisms in the Internet environment.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICE 98 Charleston SC USA
Copyright ACM 1998 1-58113-076-7/98/10...$5.00

Keywords:

Global Computation, Internet, Resource Allocation, Markets, Java

1. Introduction

There are currently millions of processors connected to the Internet. At any given moment many if not most of them are idle. An obvious and appealing idea is to utilize these idle processors for running applications that require large computational power. This would allow what may be termed "global computing" – a single computation carried out in cooperation between many processors worldwide.

Similar ideas in the context of local area networks are quite well known by now, especially due to the influence of the work on "Network of Workstations" [1]. There are several added complications, though, in the global case of cooperation over the whole Internet. First, there are major technical difficulties due to code mobility, security, platform heterogeneity, and coordination concerns. Then there is a matter of scale as the Internet is much more "distributed": The communication bandwidth is smaller, the latency higher, the reliability lower. On the positive side, the potential number of processors is huge.

A much more fundamental difference is due to the distributed *ownership* of the processors on the Internet. Since each processor is owned and operated by a different person or organization, there is no a-priori motivation for

* Supported by a grant from the Israeli ministry of science.

cooperation (why should my computer work on your problem?) Clearly a motivation for cooperation, such as payments for CPU time, must be provided by a global computing system.

The POPCORN system provides an infrastructure for such "global computation", addressing all these difficulties. It has been implemented and was operable until recently for over a year through its web site over the Internet [38]. Descriptions of the system can be found in [6, 21, 32, 37].

This paper focuses on POPCORN's approach to the last issue, that of motivating cooperation. The POPCORN system provides an online electronic market for trade in CPU-time. Buyers and sellers of CPU-time connect to it via the Internet, and the market matches buyers and sellers according to economic criteria. It should be emphasized that the buyers and sellers are computer programs (acting for humans) rather than directly humans. It seems very likely that such totally automated electronic markets will play a large role in many forms of internet cooperation (not just for CPU time), and that general mechanisms for such markets need to be developed and understood.

The design and implementation of such online electronic markets will of course draw on the vast literature available regarding real-world markets [2, 11, 12, 13, 35]. However, one should note that many differences exist. First, there are many technical issues of communication, implementation, etc. Second, the fact that the market is not intended for humans but rather for programs makes a difference. Third, in many cases, and in particular in the case of the POPCORN market, even the basic definitions of money, goods and trade need to be defined. We hope that our experiences with the POPCORN market may shed further light on several aspects of online electronic markets.

1.1 Paper Structure

Section 2 provides an overview of the POPCORN system; further information can be found in [6, 21, 32, 37]. Section 3 describes the outline of the economic notions and mechanisms which underline the POPCORN system. Section 4 provides preliminary analysis of the POPCORN markets. Section 5 describes our simulations of the POPCORN trade. In section 6 we mention some related works and section 7 outlines directions for further research.

2. An Overview of the POPCORN System

The POPCORN system provides an infrastructure for global computation over the Internet. POPCORN's basic function is to provide any programmer on the Internet with a simple virtual parallel computer. This virtual machine is implemented by utilizing all processors on the Internet that care to participate at any given moment. The system is implemented in Java and relies on its ubiquitous "applet" mechanism for enabling wide scale safe participation of remote processors.

There are three distinct types of entities in the POPCORN system:

1. The parallel program written (in Java) using the POPCORN paradigm. This program acts as a CPU-time "*buyer*". The program is written using the POPCORN programming paradigm that was designed to fit "global computing". This paradigm is described in [21, 32].
2. The CPU-time "*seller*" who allows its CPU to be used by other parallel programs (instead of standing idle). This is done as easily as visiting a web-site using a Java-enabled browser, and requires no download of code.
3. The "*market*" which serves as a meeting place and matchmaker for buyers and sellers of CPU-time.

The POPCORN programming paradigm, used by the buyer program, achieves parallelism by concurrently spawning off many sub-computations, termed "*computelets*". The POPCORN system automatically sends these computelets to a market (chosen by the user), which then forwards them to connected CPU-time sellers, who execute them and return the results. The matching of buyers and sellers in the market is dynamic, is done according to economic mechanisms, and results in a payment of the buyer to the seller.

The system is intended for coarse-grained parallelism. The computational efficiency is mostly determined by the ratio between the computation time of computelets to the communication effort needed to send them and handle the overhead. To achieve high efficiency, computelets should be relatively heavy in terms of computation time. Currently, seconds of CPU-time per computelet are a minimum, and tens of seconds seem more typical. For very large-scale computations, even hours make sense. Sample application for POPCORN include brute-force search, code-breaking, simulated annealing, genetic algorithms, and game-tree evaluation.

Detailed descriptions of the POPCORN system can be found in [6, 21, 32]. Further, and up to date, information can be found on the POPCORN web site [37].

3. A Micro-Economy of CPU time

3.1 The Goods

The first thing one must ask in such an electronic market is what exactly we are trading in. The answer "CPU time" is not exact enough since it lacks specifics such as units, differences between processors, deadlines, guarantees, etc. A basic tradeoff in answering this question is between allowing the traders very specific description of the goods, and between maintaining a small number of uniform types of goods with larger market size. Our approach has been to emphasize uniformity in the initial implementation, but building the infrastructure to allow specialization in later versions.

Our basic goods are the "JOPs" – Java Operations. This is the Java equivalent of the commonly used, though imprecise, FLOPS. Of course, there are different types of Java operations, with different costs in different implementations, so we define a specific mix of computations and use this mix as a definition. Each computelet takes some number of JOPs to execute, and the price for a computelet is proportional to the number of JOPs it actually took to compute remotely. This is measured (or actually, approximated) using a simple benchmark we piggyback on each computelet.

Our experience suggests that this mechanism works well. Still, two main disadvantages are obvious: first, the benchmark is run on the sellers' computer and this computer may cheat and report higher numbers. (Such cheating entails modification of the browser used on the sellers' side, but is still possible with some effort.) Second, it is imprecise by nature, as well as has an overhead. We have thus also provided a second type of "good" which can be traded: simply the computation of a single computelet. This does not require any benchmarking, but may be troublesome for the seller since he has no a-priory control over the computation time of the computelet. Still, this simple mechanism is very proper in many situations such as the case of a repeated buyer of CPU time, the case of "friendly" non-financial transactions, or the case where computelets' size is set administratively.

3.2 The Money

One may think of several motivations for one processor to provide CPU-time to another. They may belong to the same person or organization, one might donate its CPU-time "for a good cause", the processor may get something in return, or it may get CPU-time in return at a later time. As in real life, all of these motivations, as well as others, may be captured by the abstract notion of money. This "money" may be donated, traded, bartered, loaned, converted to other "currency", etc.

This is the approach taken in POPCORN: we define an abstract currency called a *popcoin*. All trade in CPU time is ultimately done in terms of popcoins. In our current implementation popcoins are just implemented as entries in a database managed by the market, but they can be easily implemented using any one of the electronic payment schemes. Each user of the POPCORN system has a popcoin-account, paying from it for CPU time required, or depositing into it when selling CPU time. The market automatically handles these financial aspects throughout the computation. Once this mechanism exists, all of the motivations described above are obtained by administrative decisions regarding how you view popcoins: If you want to get true payment for CPU time just provide conversion between popcoins and US$ (we do not...). If you are in a friendly situation just ignore popcoin amounts. If you want to loan CPU cycles, just buy CPU time with popcoins and at another time sell your CPU time for popcoins.

3.3 Buying and Selling CPU time

The programmer writing a parallel POPCORN application is in fact buying CPU time. Basically, the parallel program must offer a price for the computation of each computelet. The payment is executed on sellers' return of the answer to the market, and is deducted from the buyers' account in the market (which must be specified before the computation can proceed). Technically, each computation packet constructs a "Contract" object that encapsulates the offer. The contract specifies the prices offered, whether the price is per computelet or per JOP, and the market mechanism required for this transaction (see below). The contract may be hard-coded into the program; alternatively we provide a user-level tool for specifying the contract.

Selling CPU time is done as easily as visiting a page on the web with a Java-enabled browser. This page contains an applet that starts working on the sellers' computer and which repeatedly receives computelets and computes them. In the most direct form, a seller visits the market's web site, where he is asked to provide his account information (name and password). Once this information is provided, a "start computing" button starts the CPU-selling process and all popcoins earned are deposited into this account. By default, each seller simply auctions his CPU-time to the highest bidding (per-JOP) prospective buyer. Optionally, the seller may also enter his preferences for the trade, e.g. specifying pricing information (see below).

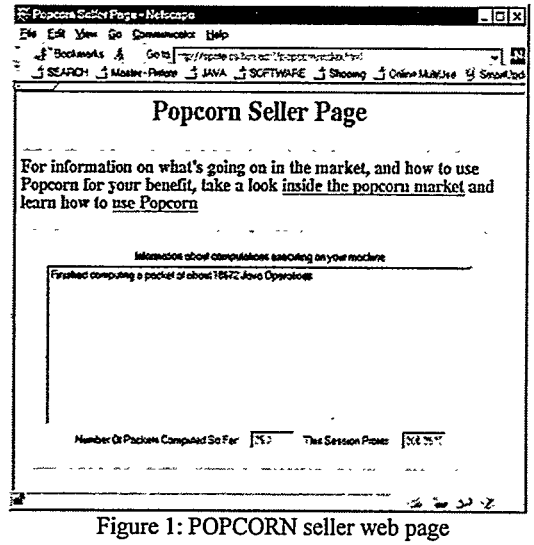

Figure 1: POPCORN seller web page

An alternative mechanism exists which does not require the seller to hold an account, or to be compensated in popcoins. In this variant a "seller" visits a web page that is of some interest to him. In addition to this interesting information, the page contains the "POPCORN logo". This logo has two main functions. First, it is an applet that receives computelets and executes them. Second, this logo explicitly

informs the user that this is going on. In this situation the seller is in fact bartering his CPU time for the information on the web page. This can be an on-line game, a picture, or any other type of information. We maintain a little "gallery" of such web-pages [36]. In effect we have a 3-way trade here: the seller provides CPU time and gets information; the page owner ("publisher") provides information and gets popcoins; and the buyers provide popcoins and get CPU time. Exact details on how to become a publisher can be found at [39].

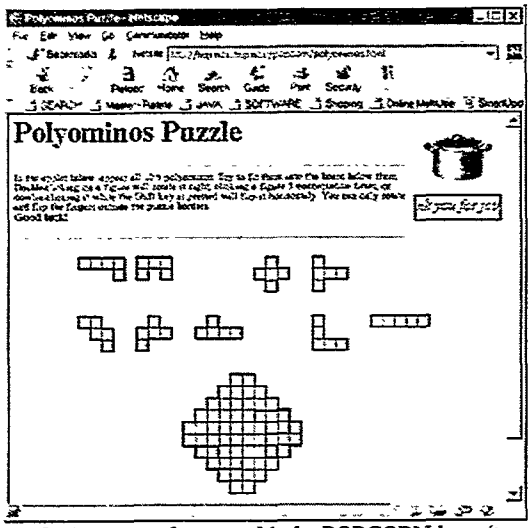

Figure 2: A game web page with the POPCORN logo (top right corner)

3.4 The Market

The most immediate function of the market is to simply serve as a well-known location which buyers and sellers come to, instead of trying to look for each other all over the Internet. (There can be many different markets, but supposedly, each is in a "well-known" location.) Obviously, this makes the market a communication bottleneck of the whole system, but as long as the computation done by each computelet is CPU-time-consuming enough relative to the market overhead, a single market can handle large numbers of buyers and sellers. The market is a trusted intermediary and is responsible for matching buyers and sellers, for moving the computelets and results between them, as well as for handling all payments and accounts. The implementation of the market involves a server that buyers can connect to as clients, as well as a set of web pages with applets embedded in them (and a supporting server) for the sellers to connect to.

The most important aspect of the market is to match buyers and sellers according to economic criteria. There are two basic design goals for the market mechanism:

1. The allocations should be economically efficient, i.e. maximize the global utility by allocating CPU-time to those with the highest utility for this time. A necessary condition for this is the second requirement.
2. The market mechanism should motivate the buyers and sellers to reveal their true utility of the CPU time (i.e. the mechanism should be "incentive compatible" [16].) This frees them from strategic considerations and, we feel, is especially important in an electronic market setting, in which the bidding is pre-programmed rather than interactively done by humans.

We have three different mechanisms currently available. They are currently handled as separate internal markets and each buyer and seller may choose which of these mechanisms he desires (in addition to choosing whether the payment is per JOP or per computelet – defaults exist for all these choices, of course). All our mechanisms are of the sealed-bid type ("fire and forget"). They thus require only a single round of communication. In addition, the mechanisms are all efficiently computable (usually requiring just a few hashing and priority queue operations per computelet.)

The first mechanism is a repeated Vickrey auction [40]. Here, the CPU-time of the sellers is auctioned among current buyers – separately for each computelet. The catch in a Vickrey auction is that the price paid by the buyer is not the one offered by him but rather the second highest price offered in the auction.

The second mechanism we provide is a simple sealed-bid double auction [15] (DA hereafter). In this case, both buyers and sellers offer a low price and a high price, as well as a rate of change. The sellers start with an offer of the high price – an offer which is automatically decreased at the specified rate, until a buyer is found (or the low price reached). Similarly, buyers start at the low price, and their offer is automatically increased until a seller is found (or the high price is reached). When a buyer "meets" a seller the buyers' computelet is sent for execution on the sellers' machine and the payment is at the meeting price. This mechanism is very dynamic and easy to define and implement.

The third mechanism we provide is a repeated Clearinghouse DA (also known as the k-DA [15]). This mechanism is similar to the simple DA, except for the fact that in each round more than one buyer-seller pair is matched. At fixed time intervals, all the sealed-bids of buyers and sealed-asks of sellers are used to calculate current demand and supply curves (the demand and supply curves are step functions). These curves are intersected, the equilibrium price is calculated, and the single price is used for all the transactions of that round. We take the trading price to be the middle of the interval of possible clearing prices (i.e. we use a k-DA with $k = 1/2$).

4. Towards an Analysis of the POPCORN Market Mechanisms

The basic question about these market mechanisms is whether they work well in terms of finding an economically efficient matching of buyers and sellers, in a the dynamic situation of the Internet. In addition, the computational efficiency of reaching this matching is of importance, as to enable the market to function online under high loads. From an economic point of view, our main interest is in checking whether the gains-from-trade that are generated by our mechanisms' allocations are maximized. The results of our analysis of the market mechanisms are quite sensitive to the exact model of the buyers and sellers as well as to the economic utility placed on time and to the information available to the players. Some results from game theory and mechanism design literature [16] are directly or indirectly applicable, yielding both positive and negative results. In this section we describe these analytical steps towards analysis, while in section 5 we describe results of our simulation, using the same theoretical framework.

One way of analyzing a POPCORN market is to view its rules as defining a multiple-round game of incomplete information, in which the number of players may change as time progresses and the players' actions in one round affect the future rounds. Each trade-round at the market corresponds to one round in the game. The players of the game are the selfish buyers and the sellers, who seek to maximize their private utility. The game that fully corresponds to the POPCORN trade is difficult to analyze, and we make some simplifying assumptions in our discussion below.

In order to model a buyer we assume that a buyer has a *type* V_i^0, drawn from a probability distribution F. The buyer's type corresponds to his *valuation of a JOP at time 0*, and once a buyer's type was determined at time 0, his valuation of a JOP (in terms of time 0) decays as the calculation of that JOP is done further into the future. Formally, buyer i's valuation of the calculation of a JOP at period t in terms of period 0 is $V_i^t = \alpha^t V_i^0$, where $0 \le \alpha \le 1$ is a discount factor common to all buyers. We also assume that if a buyer cuts a deal at time t, and buys a computation of L *jops*, at a price p, he enjoys a utility of $L(V_i^t - p)$. A buyer generates a stream of orders, beginning with a *'connect'* order, followed by *'bid'* orders, and finally a *'disconnect'* order.

Similarly our model of a seller states that a seller is defined by a type e_i, drawn from a probability distribution G. The seller's type corresponds to his *expenses per a unit of time.* Another attribute of the seller is his machine's speed of z_i *jops/second*, which is drawn from a probability distribution H. We assume that if a seller cuts a deal to sell a computation of L *jops*, at a price p, he enjoys a utility of $L(e_i/z_i - p)$. A seller generates a stream of orders, beginning with a *'connect'* order, followed by one or more *'ask'* orders, and finally a *'disconnect'* order. We assume that both buyers and sellers know their type a-priori (i.e. we assume that the *independent private value model* [26] holds).

We also assume that the seller publishes no information about his client's computer, that the auction rounds are held at discrete points in time and that all the constraints imposed by the POPCORN architecture are satisfied.

4.1 The Repeated Vickrey Auction Mechanism

A well-known result in auction theory is that it is a dominant strategy for a buyer in a Vickrey auction to bid his true valuation of the good being auctioned [40] and that the Vickrey auction is a *direct-revelation mechanism* [16]. Thus, the outcome of a single round of the Vickrey auction satisfies the strong criterion of *dominant strategy equilibrium.* When the players play their equilibrium strategies the outcome of a single round is efficient – the buyer who posses the highest valuation for the good wins it.

However, our Vickrey-based mechanism is executing a Vickrey auction repeatedly and the properties of the single round do not necessarily hold. Despite this theoretical possibility, we do not know of *practical* ways, by which programs can gainfully employ strategic reasoning in the context of the POPCORN market. If we do assume that the buyers and sellers are truth-tellers (i.e. non-strategic buyers and sellers) and that the sellers are homogenous, then we can show that the mechanism maximizes the welfare of the economy. The proof is straightforward and we do not bring it here, please refer to [33] for details.

Our assumptions about the homogeneity of the buyers and sellers are not reasonable in Internet-based markets. The machines connected to the Internet differ in their computation power, connection quality, the costs of connection differ from one ISP to another, etc. If we relax our assumption that the sellers' costs are equal or that the sellers' machines are identical, the efficiency breaks. The reason is obvious: the mechanism chooses the sellers arbitrarily, based solely upon their arrival time. Relaxing the assumption that the buyers and sellers know their valuations also leads to inefficient outcomes. On the other hand, the efficiency result holds when we assume asymmetric buyers [33]. Taking these issues into account will naturally complicate the system and may be a subject for further research.

4.2 The Double Auction Mechanisms

We have seen that in the general case the repeated Vickrey mechanism, described in the previous section, is socially inefficient. One reason for inefficiency is that it does not introduce competition among the sellers. A possible remedy is to consider double auctions.

4.2.1 A Simple Double Auction

The simple DA resembles the simultaneous execution of two *"first-price sealed bid auctions"*[26, 29]. Very much like in the case of first price sealed bid auction [25, 26, 28, 29, 30], there is no dominant strategy equilibrium in the simple DA. We are left with a weaker criterion of *Bayesian Nash equilibrium* [16]. In addition, it is not a direct-revelation mechanism, and each round of our auction is not necessarily Pareto efficient. We return to the simple DA in section 5.

4.2.2 The Clearinghouse Double Auction

In [34] it is shown, for a single round of k-DA, that if each buyer requests a single good and each seller possess a single good then as the number of buyers and sellers grows, the incentive for buyers and sellers to falsely report their types diminishes. In this case the gains from trade converge to the maximal possible value. However, this result does not hold when each trader is interested in multiple units. Furthermore, multiple rounds may lead to inefficient results, when the mechanism is executed against an adversary who seeks to minimize the gains from trade. It can be shown that the repeated clearinghouse algorithm is not α - competitive for any constant α [33].

Mendelson [27] analyzes a mechanism very similar to ours, assuming that the bids and asks are uniformly distributed over some interval, that the arrival of orders from buyers and sellers are governed by identical Poisson processes and that the players are not strategic (they report their true valuations). Mendelson's analysis implies that "we cannot be very far from that price that maximizes the gains from trade most of the time". At the time of writing these lines, we have insufficient empirical data for supporting or refuting Mendelson's assumptions.

5. Trade Simulation

In this section we check the market's behavior and characteristics by simulating the trade that results from the dynamic arrival and departure of various buyers and sellers. There are many empirically open questions, of which we consider only a few. Perhaps the most interesting issues are the allocation's efficiency in a dynamic environment and the price response to supply and demand shocks. In addition, we check the markets adaptivity and the relation between a seller's (or buyer's) offer and his probability of executing the trade.

We conducted most of the experiments in a constrained environment. In this environment, the buyers and sellers do *not* behave strategically, each buyer has a single computelet to compute, and each seller wishes to compute a single computelet. Overall, in the very simple settings of our experiments, the results are promising: The markets present the expected behavior in terms of price trends, the efficiency of their allocation is surprisingly high, the speed of the economy's adaptation is quite high, the prices are relatively stable, and reservation prices have the expected effect.

5.1 A Test-bed for Market Simulations

In order to simulate the operation of the various POPCORN market mechanisms, we have implemented a simulations test-bed. The implementation provides much flexibility in varying the parameters that influence the trade. Figure 3, below, provides a schematic description of the simulation test-bed architecture.

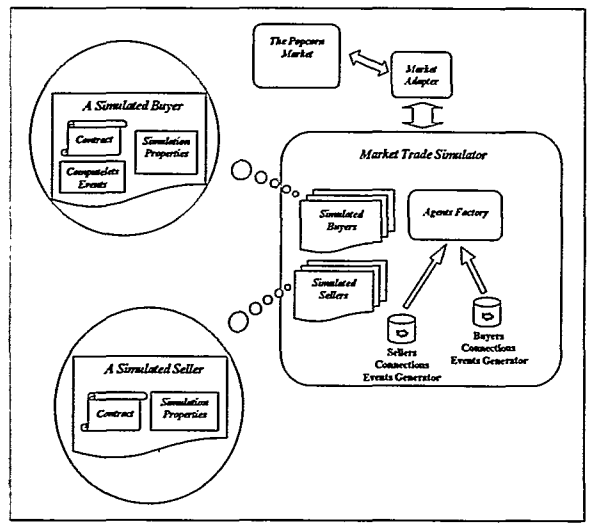

Figure 3: The Simulation Test-bed Architecture

The heart of the simulation test-bed is a *Market Simulator* object, which coordinates among the various objects. The Market Simulator is connected to the POPCORN market through a *Market Adapter*, which replaces the real POPCORN communication layer. This enabled us to simulate the trade without requiring hundreds of computers. The Market Simulator contains an *Agents Factory*, which creates the buyers and sellers according to stochastic processes of some kind. We used a *Poisson Process Generator* with configurable parameters in our simulations.

Upon creation, a *Simulated Buyer* (or *Seller*) reads his characteristics from a file. Most of the parameters that it reads define a probability distribution, from which the *Simulated Buyer* draws the real parameter to be used during the simulation. One important characteristic of the *Simulated Buyer* is his computelet spawning process, which is also governed by a stochastic event-generator. The computelet length is drawn from a uniform probability distribution. Another important characteristic of a buyer is his valuation for the good and the derived bidding strategy.

A strategy is defined by an open bid, bid increment rate, and maximum bid. *Simulated Sellers* are very much like simulated buyers. Their important attributes are their ask, selling strategy, machine speed, and lifetime period.

5.2 Simulations and Results

5.2.1 Price response to changes in the relative supply and demand

In this set of simulations, we measure the price as a function of the supply, while holding the demand (stochastically) fixed at some level. By collecting a large number of such samples, we are able to map the average price curve. The environment is consisted of truth-telling buyers and sellers, where each buyer wishes to compute a single packet of random size and each seller wishes to compute a single packet. All the sellers' machines are identical. Buyers' packets valuations and sellers' costs are drawn uniformly from the interval $[0,30]$. The Poisson process that determines the arrival of buyers remains constant, with λ fixed at 0.001 (the time units are milliseconds). The Poisson process that governs the arrival of sellers is different in each simulation and thus we achieve the changes in the supply. In a typical simulation, we used a few hundreds simultaneously-executing agents.

We have run forty simulations for each mechanism, three times for each value of the supply process lambda. All measurements were made during a steady state phase (i.e. when the percent of fulfilled requests out of the total number of requests was stable). It was reached after about five minutes of simulation. Two examples for convergence to steady state are given in figures 4 and 5.

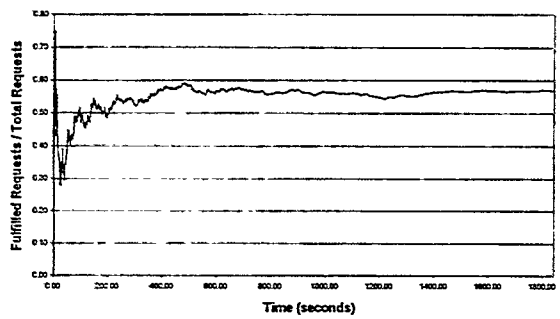

Figure 4: convergence to steady state in the simple DA

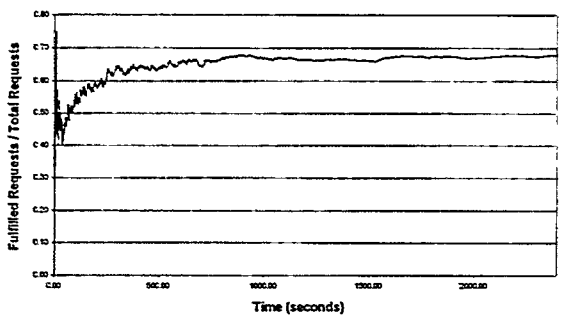

Figure 5: convergence to steady state in the Vickrey auction

The average duration of a simulation was thirty minutes. We summarize all price information in figure 6. Relative supply is on the X-axis and the resulting average price at steady state is on the Y-axis. In general, these results comply with our intuition. It is evident that an increase in the supply results in a decrease in the average price. Consider, for example, the clearinghouse mechanism. In this case, increasing the supply process lambda results in the flattening of the supply curve in every round. Since the prices are determined (in this case) by finding the crossing point of the demand and supply curves, for each value of lambda we get a different point on the expected demand curve.

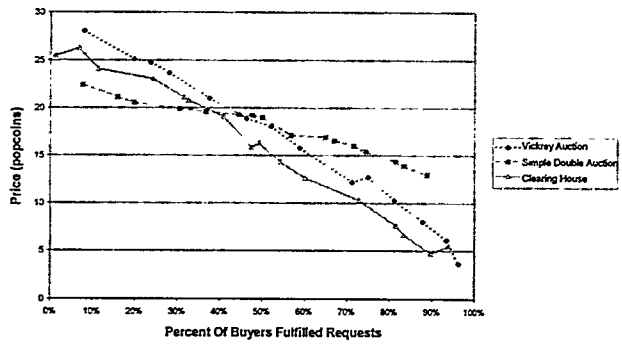

Figure 6: prices under different buyer/seller ratios

5.3 Social Efficiency of the Allocations

In order to evaluate the economic efficiency of these allocations, we compare them to the *optimal-offline* mechanism. That is, we compare the generated welfare of POPCORN's *online* mechanisms to that generated by a perfect mechanism that, *in advance*, has *all* the information about the buyers' and sellers' orders. The quotient of these two numbers represents the relative efficiency of the mechanism. This comparison is "unfair" towards the POPCORN online mechanisms as they should be compared with the best *online* mechanism that bases its decisions on the information that is available at runtime. Nevertheless, the comparison with the optimal offline allocation is still enlightening.

Figure 7 shows that the mechanisms' efficiency is quite high. In the case of the clearinghouse mechanism, it is very high with the average of 96%. We also see that the simple DA and Vickrey do not achieve such good results. The source of this difference is sellers with high costs that are matched to buyers instead of sellers with lower costs, who arrived later. When the lower-costs sellers arrive, it is too late, since the relevant buyers have already left. This effect has lower influence over the clearinghouse because in this mechanism the trade is executed in fixed time intervals.

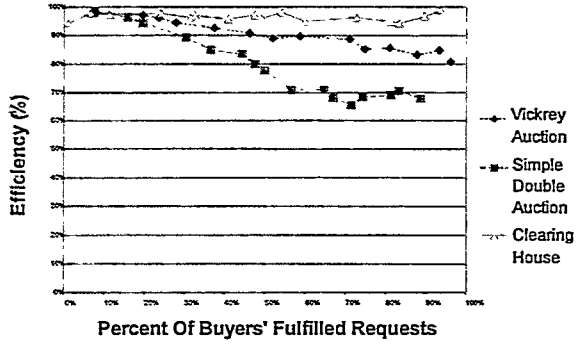

Figure 7: economic efficiency under different buyer/seller ratios

5.4 Price Stability

For a particular mechanism, the standard deviation of prices remains stable at various supply levels. This standard deviation should be considered in conjunction with the price level and we take the quotient of the measured standard deviation with the average price level. We can use these measurements to compare the two double auctions: The average value of the quotient in the clearinghouse market is 0.147, and in the simple DA it is 0.33. Thus, the clearinghouse is much more stable. Figures 8, 9 below depict the price behavior in a typical run of two mechanisms.

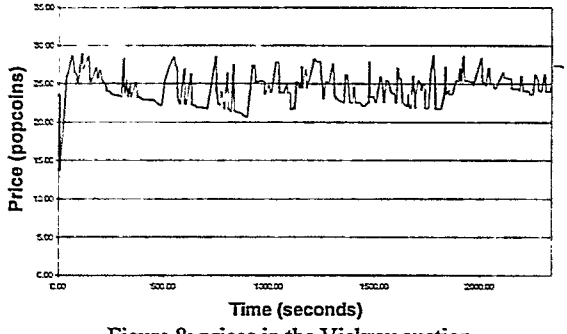

Figure 8: prices in the Vickrey auction

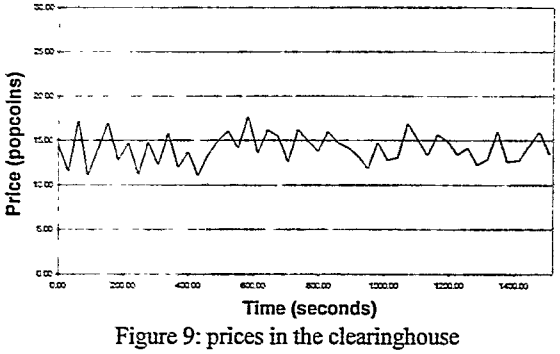

Figure 9: prices in the clearinghouse

5.5 The Role of Reservation Prices

Normally, a higher buyer offer results in a higher probability of buying. Symmetrically, a lower ask price results in a higher probability of selling. This is verified for our markets and figures 10, 11 depict this property for the clearinghouse market. Note that buy offers that are lower than a certain threshold, or asks that are higher than a threshold, never execute. We obtained similar results for the repeated Vickrey auction and the simple DA.

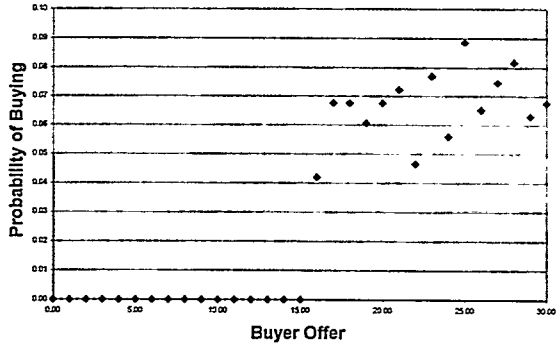

Figure 10: the probability of executing vs. the bid in the Clearinghouse auction

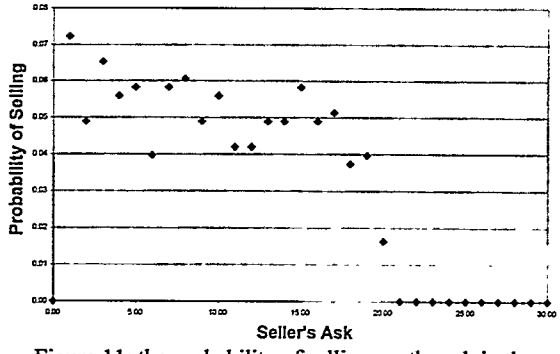

Figure 11: the probability of selling vs. the ask in the Clearinghouse auction

5.6 Other Market Properties

In the above simulations, the demand was held fixed, and the supply varied. Qualitatively, we got similar results, in the case of demand shifts. Similarly, we got the same results when buyers and sellers are interested in more that a single packet and when the sellers machines are heterogeneous.

6. Related Works

The basic idea of "stealing cycles" on local networks is well known, and is the basis for "Network of Workstations" [1] and many related projects. The same idea over the Internet, with its different considerations, is much less developed. Only recently, with the availability of the Java language, has a general mechanism for global computing been possible. Several general systems using Java have been designed, usually concentrating on a single aspect of

global computation [3, 5, 9]. These systems do not base their operation on economic principles, however.

The idea to use economic techniques, models, and intuitions, for solving a computational problem and in particular for solving allocation problems has gained popularity in recent years [4, 8, 10, 14, 17, 18, 20, 22, 41]. Similarly, the allocation of network resources and the allocation of resources over the Internet received much attention [10, 17, 19, 20, 22, 23, 31], but we are unaware of systems that allocate CPU time over the Internet and are market-oriented. The systems that allocate CPU time are generally designed to be used in LANs. For example, Spawn [41], Enterprise [24], and Challenger [7] may not be applicable to the Internet due to their communication requirements.

7. Directions for Further Research

We consider extending this work in the following directions:

1. A comprehensive efficiency analysis of the double auction is still needed. We did not conclude with a strong efficiency result even for an unrealistic model and only showed evidence for the mechanism's efficiency.
2. An obvious extension to our simulations is to experiment with strategic buyers and sellers. Our early experiments with buyers and sellers that shade their bids and asks hint to results similar to ours in the clearinghouse.
3. POPCORN was designed in such a way that we can replace parts of it. It is easy to plug-in a matching mechanism that bases its operation on more conventional non-economic-based load-balancing algorithms. A comparison with such mechanisms should be carried out.
4. We view POPCORN as a case study from which we learn on a larger family of economic-oriented computational systems. The same markets may be used for other goods. This, however, may require the definition of other billing schemes.
5. The central market is a bottleneck and the system does not scale. When connected to a 10 Megabit LAN it can handle up to 70 buyers and sellers. This is not good enough in the context of the Internet. We consider replacing the single market with a network of cooperating and competing markets.

8. References

[1] Anderson, Thomas E., Culler, David E. and Patterson, David A., "A Case for Networks of Workstations: NOW", IEEE Micro, Feb 1995, http://now.cs.berkeley.edu.

[2] Arrow, K., and Hahn F., "General Competitive Analysis", North Holland Publishing Co., Amsterdam, 1978.

[3] Baratloo, A., Karaul, M., Kedem, Z., and Wyckoff, P. "Charlotte: Metacomputing on the Web". Proceedings of the 9th Conference on Parallel and Distributed Computing System, 1996.

[4] Bogan, N.R. "Economic allocation of computation time with computation markets". Master's thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, May 1994.

[5] Brecht, T., Sandhu, H., Shan, M. and Talbot, J. "ParaWeb: Towards Word-Wide Supercomputing", Proceedings of the Seventh ACM SIGOPS. European Workshop, Connemara, Ireland, pp. 181-188. September 1996.

[6] Camiel, N., London, S., Nisan, N. and Regev, O. "The POPCORN Project – An Interim Report, Distributed Computation over the Internet in Java", Proceedings of the sixth international World Wide Web conference (CD version), 1997, http://www.cs.huji.ac.il/~popcorn/documentation/www6/poster.html.

[7] Chavez, A., Moukas, A., and Maes, P.. "Challenger: A Multi Agent System for Distributed Resource Allocation". Proceedings of the First International Conference on Autonomous Agents. Marina del Ray, California, February 1997.

[8] Cheng, J.Q., and Wellman, M.P. "The WALRAS algorithm: A convergent distributed implementation of general equilibrium outcomes". Computational Economics, to appear.

[9] Christiansen, B., Cappello, P., Ionescu, M.F., Neary, M.O., Schauser, K.E., and Wu, D. "Javelin: Internet-Based Parallel Computing Using Java", 1997 ACM Workshop on Java for Science and Engineering Computation, June 1997

[10] Cocchi, R., Estrin, D., Shenker, S., et al. "Pricing in computer networks: Motivation, formulation, and example". IEEE Transactions on Networking 1(6): 614-627, 1993.

[11] Debreu, G. "Theory of Value", John Wiley Sons, New York, 1959.

[12] Domowitz, I. "Automating the Continuous Double Auction in Practice: Automated Trade Execution Systems in Financial Markets". The Double Auction Market Institutions, Theories, and Evidence, Friedman D. and Rust J. (Eds.), Proceeding of The Workshop On Double Auction Markets, Santa Fe, New Mexico, June 1991.

[13] Eitman, W.J., and Eitman, S.C. "Nine Leading Stock Exchanges", Ann Arbor, Michigan International Business Studies, The University of Michigan, 1968.

[14] Ferguson, D., Yemini, Y., and Nikolaou, C. "Microeconomic algorithms for load balancing in distributed computer systems". Eighth International Conference on Distributed Computing Systems, 1988.

[15] Friedman, D. "The Double Auction Market Institution: A Survey". The Double Auction Market Institutions, Theories, and Evidence, Friedman D. and Rust J. (Eds.), Proceeding of The Workshop On Double Auction Markets, Santa Fe, New Mexico, June 1991.

[16] Fudenberg, D., and Tirole, J. "Game Theory". Forth Edition, MIT Press 1995.

[17] Gupta, A. Stahl, D.O., Whinston, A.B. "Managing The Internet as an Economic System". University of Texas at Austin, Research Papers, 1994.

[18] Huberman, B. "Computation as Economics". Second International Conference in Economics and Finance, 1996.

[19] Korilis, Y. A., Lazar, A. A., and Orda, A. "Architecting Noncooperative Networks". Journal on Selected Areas in Communications 13(7): 1241-1251, 1995.

[20] Lazar, A.A. and Semret, N. "Auction for Network Resource Sharing". CTR Technical Report: CU/CTR/TR 468-97-02, Center for Telecommunications Research, Columbia University, February 1997.

[21] London S. The Popcorn Tutorial, http://www.cs.huji.ac.il/~popcorn/developer/tutorial/index.html.

[22] MacKie-Mason, J. K., and Varian, H. R. "Pricing congestible resources". IEEE Journal on Selected Areas in Communications 13(7): 1141-1149, 1995.

[23] MacKie-Mason, J. K., and Varian, H. R. "Pricing the Internet". Public Access to the Internet Ed. B. Kahin, and J. Keller. MIT Press. 269-314, 1995.

[24] Malone, T. W., Fikes, R. E., Grant, K. R., et al. "Enterprise: A Market-like Task Scheduler for Distributed Computing Environments". The Ecology of Computation Huberman, B. A. (Ed.). North-Holland. 177-205, 1988.

[25] Maskin E., and Riley, J. "Existence and Uniqueness of Equilibrium in Sealed High Bid Auctions". Mimeo, UCLA, March 1986.

[26] McAfee P., and McMillan J. "Auctions and Bidding". Journal of Economic Literature, 25, pp. 699-738, June 1987.

[27] Mendelson, H. "Market Behavior in a Clearing House". Econometrica 47, pp. 61-74, 1982.

[28] Menezes, F., and Monteiro, P. "Sequential Asymmetric Auctions With Endogenous Participation". Economics Working Paper Archive, http://econwpa.wustl.edu/wpawelcome.html, ewp-mic/9402001, 1994.

[29] Milgrom, P. "Auctions and Bidding: A Primer". Journal of Economic Perspectives, Vol. 3 (3), pp. 3-22, Summer 1989.

[30] Milgrom, P., and Weber, R. "Distributional Strategies for Games with Incomplete Information". Math Operations Research, 10, pp. 619-632, November 1985.

[31] Mullen, T., and Wellman, M.P. "A simple computational market for network information services". First International Conference on Multiagent Systems, San Francisco, CA, MIT Press 1995.

[32] Nisan, N., London, S., Regev, O., and Camiel N. "Globally Distributed Computation over the Internet – The POPCORN Project", proceedings for the 18th International Conference on Distributed Computing Systems (ICDCS'98), Amsterdam, The Netherlands, 1998, to appear.

[33] Regev, O. "Economic Issues in CPU Sharing System for the Internet". Master's Thesis, The Institute of Computer Science, The Hebrew University, Jerusalem, 1998.

[34] Rustichini, A., Satterthwaite, M., and Williams, S. "Convergence to Price-Taking Behavior in a Simple Market". D.P. 914, Center for Mathematical Studies in Economics and Management Science, December 1990.

[35] Spray, D. E. "The Principle Stock Exchanges of the World: Their Operation, Structure, and Development", International Economic Publishers Inc., Washington D.C., 1964.

[36] The Popcorn Gallery, http://www.cs.huji.ac.il/~popcorn/gallery/index.html

[37] The POPCORN homepage, http://www.cs.huji.ac.il/~popcorn.

[38] The Popcorn Market homepage, http://www.cs.huji.ac.il/~popcorn/use.html.

[39] The Popcorn publisher homepage, http://www.cs.huji.ac.il/~popcorn/publisher/index.html

[40] Vickrey, W. "Counterspeculation, Auctions, and Competitive Sealed Tenders". Journal of Finance, 16, pp. 8-37, March 1961.

[41] Waldspurger, C.A. et al. "Spawn: A Distributed Computational Economy". IEEE Trans. on Software Engineering, Vol. 18, No. 2, Feb 1992.