[CH] Cheriton and Harty [5] have proposed an operating system extension that would allow application programs to have greater control over their own memory use. The mechanism is intended to serve as a way for memory scheduling to be devolved to applications in order to promote efficiency, rather than to increase fairness. Applications would be granted an allowance of "money" with which to purchase uninterrupted use of blocks of memory for a period of time. The application programmer can be expected to know the program's pattern of memory use better than any general purpose, central scheduling algorithm, hence the application could purchase the size and duration of memory residency it requires to proceed with greatest efficiency. Simultaneously executing applications would not enter into economic competition with one another because the system would allocate them money on a conventional priority basis, limit the ability to save, and hold resource prices fixed, to promote planning.

[DN] Dafermos and Nagurney [8] have presented fast equilibration algorithms for a restricted problem in which m producers and n consumers of the same commodity are to be matched. There is no initial endowment of goods: instead, each producer and consumer has a linear supply or demand function that indicates its production or consumption at a given price. The equilibration problem is to find prices for each producer and consumer and a set of shipments such that supply balances demand. Their algorithm starts with a feasible set of shipments and uses iterative relaxation to improve the situation of each producer or consumer in turn, until equilibrium is achieved. Linearity of production and consumption yields a closed form expression of the restricted equilibrium used in relaxation. Feasibility of shipment set is preserved at all stages. No attention was given to distribution of the algorithm; it does not appear to be easily distributable. This paper was inspired by applications in transportation, and is part of a broader operations research literature on similar algorithms.

[F1] Ferguson *et al.* [11] designed a market method tailored to calculate a static price equilibrium that optimally allocates link capacity to virtual circuits. In their Flow Control Economy the link edges are the goods and virtual circuits are the consumers. Each virtual circuit is endowed with a budget from which to purchase link capacity, and has a preference function over edge allocations that first seeks to satisfy a minimum throughput, then tries to reduce worst-case average delay. They are able to define the preference functions such that a unique equilibrium price vector exists. Their tâtonnement equilibration algorithm starts with an arbitrary price vector then iteratively adjusts prices according to excess demand until convergence, whereupon the link allocation is determined.

[F2] Ferguson *et al.* [10] studied use of a market algorithm to dynamically distribute read/write copies of data objects in a network. They modeled a transaction system in which jobs are initially allocated a budget from which to pay for data access. Processor nodes in the network sell access to objects they retain locally, and subcontract access to other objects. Read copies are leased by processors, while the single write copy is periodically auctioned. The cost of a read operation depends on its cost at the nearest node with a copy of the data. The cost of a write operation depends on the location of the write copy and the number of read copies outstanding. Nodes decide what read copies to lease and how much to bid for the write copies on the basis of expected demand for access, so as to maximize profitability. The scheme is rather complex and contains a number of heuristics. A simulation of a small ring network holding 10 data objects showed that the algorithm achieved better response time for transactions than two unsophisticated, alternative distribution schemes.

[HC] Huberman and Clearwater [20] describe an experimental multiagent system used to control the flow of air in a building's heating/cooling ducts. The system is structured as a collection of bidding agents representing regions controlled by a single thermostat who communicate with a central auctioneer. The agents participate in an ongoing series of double auctions whose clearing prices are determined by the auctioneer, and whose outcomes determine the positions of the air flow control valves in the ducts, until the next auction. Nevertheless, the algorithmic characteristics of the system appear analogous to the auctionless, quasi-market method proposed by Ho [18]. Each agent decides whether to buy or sell air based not on its local degree of comfort, but on the ratio of its local control error (measured temperature − thermostat setting) to global control error: in hot weather, agents faring worse than average bid for more cool air, while others sell. Agents are not allowed to accumulate money, but instead are granted a bid allowance for each auction that is a linear function of flow valve position (agents with closed valves are given more "money" to bid). Consequently, agent bids are a non-linear function of local error, global error, and local valve position. Successful "buyers" and "sellers" are directed to open and close their flow control valves by some corresponding degree. The effect is to balance the degree of discomfort when there is a shortfall of cool air entering the system on a hot day. The authors report that their system reduced standard deviation of control error below that achieved by a conventional integral control system.

[Ho] Ho *et al.* [18] present a framework for what they call center-free resource allocation algorithms. They address the simple problem in which each node in a network has a utility function f_i. The sum $\sum f_i(k_i)$ is to be maximized by adjusting the allocations k_i of a single, uniform and continuously divisible good to each node, subject to the constraints that all allocations are non-negative and $\sum k_i = K$. They note that the maximizing allocation is characterized by two conditions: allocation of the total amount K, and equality of marginal utility. Ho *et al.* note that preliminary allocations under an iterative convergence technique such as tâtonnement are not guaranteed to be feasible, and suggest use of a resource-directed approach in which the full amount K is always allocated, and only the distribution is adjusted until equilibrium is achieved. The resource-directed approach has the additional advantage of being distributable over a range of granularities in a way that trades off communication costs for speed of convergence. The basic scheme is that each node calculates its own marginal utility at each iteration, then communicates with its neighbors and redistributes some of the good so that nodes with below-average marginal utility get less, and the rest get more. The algorithm is essentially cooperative in that nodes freely sacrifice local utility for improved global utility. Ho *et al.* show that these algorithms are guaranteed to converge while monotonically improving total utility.

Kurose and Simha [22] applied the work of Ho *et al.* to the allocation of a file resource in a distributed system, and extended it by proposing an algorithm that uses the second derivative of utility to speed convergence.

[LS] Waldspurger, one of the authors of the Spawn system, has devised a light-weight, proportional-share resource management technique called *lottery scheduling* [40]. The technique dynamically equilibrates supply and demand, but does away with the need for any explicit price adjustment or billing mechanism. Use of a resource, say a CPU, is assigned for a limited time to a job selected stochastically from the eligible pool. Jobs are allocated a right to the use of a resource in the form of a lottery ticket whose expected allocation rate is proportional to its denomination and the sum of extant tickets for that lottery. In other words, the holder of a ticket with a weight of 10 has a 10% chance of getting the resource during a time period when a total ticket weight of 100 has been distributed, and a 50% chance when the total ticket weight is 20. Hence a job's chance of winning is dynamically adjusted as tickets are created and destroyed. Tickets have some additional benefits such as being transferable (*e.g.,* from a client to a server).

[M] Mariposa [35, 36, 33] is a distributed database project whose design incorporates market methods for distributing sub-parts of queries to servers and also for managing storage. The overall scheme is roughly similar to [10], but much more ambitious. It includes contracts with penalties; provision for open bidding, as well as cheaper-overhead methods of finding sub-contractors such as advertisements and coupons; and a notion of spheres of influence and discounting of currencies. At the time this is written, the Mariposa group has not yet reported on experience with a running version.

[S] In the Spawn [39] system idle cycles on a workstation LAN were harvested to run highly-distributable Monte Carlo simulation programs by auctioning gross time slices (approx. 60 sec.) on unused workstations. Agents representing processes would bid for time slices according to their needs and budget, then the winner would run. Provision was made for splitting a process into sub-processes to take advantage of multiple workstations, hence the name. Funding flowed dynamically from parent to child processes, in the form of an income. Waldspurger *et al.* found that Spawn jobs tended to be allocated time in proportion to their funding; that approximate price equilibria were achieved for heterogeneous machines that corresponded to their relative value (as programmed into the bidding functions), and that jobs tended to spread smoothly over the network, with time.

[T] Carl Ellison [9] reported on early use of a market-inspired technique for improving the scheduler of the TENEX time sharing system. The existing scheduler did a poor job of balancing a mix that included large, politically important batch jobs together with small, interactive student jobs, all on a machine with extremely limited core memory. The new scheduler started with a new definition of fairness: All users are assigned an income rate (in an internal currency used only for this purpose), on the basis of project administration priorities. This income accrues only while a job is pending, and one user's income is divided among all his concurrent jobs. Each job dynamically pays for its use of CPU and memory, and cannot use any more than it can afford. The practice of charging for memory residency as well as CPU has the effect of charging jobs not only for the resources they use, but also for their prevention of other jobs' use, since memory residency by a job that is waiting for a page to load can block the progress of other jobs. In place of a priority queue a balance set of solvent jobs is maintained, and insolvent jobs are kicked out to languish in secondary storage until their income replenishes their account. Solvent jobs in the balance set are treated with rough equivalence. The balance set is maintained at a reasonable size by periodically adjusting the prices of CPU and memory according to a simple difference equation, to keep demand within the bounds of supply. Ellison reported very good results from this simple mechanism.

[W] Wellman appears to have been the first computer scientist to recognize that general equilibrium can be treated as a general solution technique for computational problems and put it into practice. He is exploring an approach he calls market-oriented programming [43] and has designed a general tool named WALRAS towards that end. A program is specified in WALRAS by defining a set of consumers and producers and their utilities, technologies and endowments. WALRAS provides template definitions of standard agents and functions so that much of this task can be accomplished just by specifying parameter values. The WALRAS environment then solves the problem by iteratively applying a built-in price adjustment technique until convergence. Wellman's interests are in distributed AI and he suggests that the market-oriented programming approach is most appropriate in a distributed environment. Therefore he intentionally chose a tâtonnement-like equilibration technique for WALRAS that is potentially distributable. The algorithm is described in some detail in [4] along with proofs of its convergence. Wellman has studied applicability of market programming to multicommodity flow [41], configuration design [42], and location of servers in a network [28].

Appendix B: A General Equilibrium Formulation

A simple economic model for general equilibrium analysis is given as an illustration for understanding the conditions for existence of equilibria in such models. The formulation is essentially that given by Shoven & Whalley [32].

We assume the existence of n goods, whose market prices are given by the vector $\mathbf{p} = \langle p_1, \ldots, p_n \rangle$. All prices are non-negative: $p_i \geq 0$. Market demand for each good is given by a function over the whole price vector, $\xi_i(\mathbf{p})$. These demand functions are assumed to be homogeneous of degree zero, so prices can be arbitrarily normalized. The initial endowment of each good present in the economy before production is given by w_i.

Production is modeled by k "constant-returns-to-scale" (linear) activities. These activities are represented by the $n \times k$ matrix A, where entry a_{ij} indicates the amount of good i involved in production activity j at unit intensity. A negative sign indicates that the good is an input (resource), while a positive sign indicates that the good is an output (product).

$$A = \begin{bmatrix} -1 & 0 & & 0 & a_{a,n+1} & \cdots & a_{1,j} & \cdots & a_{a,k} \\ 0 & -1 & & 0 & . & & . & & . \\ . & 0 & \cdots & . & . & & . & & . \\ 0 & . & & -1 & a_{n,n+1} & \cdots & a_{n,j} & \cdots & an,k \end{bmatrix}$$

The first n activities are "slack" activities that represent the possibility of free disposal of an excess of each good. The vector $\mathbf{y} = \langle y_1, \ldots, y_n \rangle$ gives the level of each activity. These activities are nonreversable: it is not possible to produce inputs from outputs, hence each $y_i \geq 0$.

A general equilibrium in this model occurs at a price vector $\mathbf{p}^*$ and a production activity vector $\mathbf{y}^*$ characterized by two conditions:

(i) demand for each good equals its supply,

$$\xi_i(\mathbf{p}^*) = \sum_{j=1}^{k} a_{ij} y_j^* + w_i;$$

(ii) no production activity makes a profit, and those in use break even,

$$\sum_{i=1}^{n} p_i^* a_{ij} \leq 0 \quad (= 0 \text{ if } y_j^* > 0).$$

(In a pure exchange economy without production the equivalent condition for equilibrium is just $\xi_i(\mathbf{p}^*) \leq w_i$, where equality is strict when $p_i > 0$.)

The market demand functions ξ_i represent aggregate demand of consumers and producers. Consumer demand is a consequence of each consumer's utility function over consumption sets, given the assumption that consumers maximize utility. Producer demand is a consequence of each producer's production technology (the production functions available to it), given the assumption that producers seek a level of activity that maximizes total profit.

We use $\mathbf{y}^l$ to denote the production activity levels of the lth producer, and Y^l to denote the set of all such feasible vectors. The profit maximizing assumption means that the correspondence between actual production and prices is defined $Y^l(\mathbf{p}) = \{\mathbf{y}^l | \mathbf{y}^l \in Y^l$ maximizes $\sum_{i=1}^{n} p_i y_i^l\}$.

We use x_i^m to denote the consumption of good i by consumer m, and X^m to denote the set of all such feasible vectors. U^m denotes the utility function of consumer m. The correspondence between consumer demand and utility is defined $X^m(\mathbf{p}) = \{\mathbf{x}^m | \mathbf{x}^m \in Y^l$maximizes $U^m(\mathbf{x}^m)$ subject to the constraints that cost of $\mathbf{x}^m$ does not exceed the wealth of consumer m at current prices and profit levels $\}$. (By definition, all production profits are assumed to be distributed among consumers.)

In this model, the following conditions have been shown sufficient for an equilibrium price vector to exist.

1. Each production set Y^l is convex and closed.

2. $0 \in Y^l$ (inactivity is feasible).
3. Y^l is bounded from above (for some B, $\mathbf{y} \leq B$ for all $\mathbf{y} \in X^l$).
4. Each consumption set X^m is convex and closed.
5. $U^m(\mathbf{y}^m)$ is continuous and strictly quasiconcave (the "at least as desired as" set is convex; this condition implies that X^m is convex).
6. $w^m > \xi^m$ for some $\xi^m \in X^m$ (each consumer begins with an endowment of goods that is strictly greater than some bundle in its feasible consumption set).
7. X^m is bounded (no consumer can consume more than a finite amount of each good).

Slightly different conditions are possible, particularly for a different model formulation.

References

[1] K. J. Arrow and G. Debreu. Existence of equilibrium for a competitive economy. *Econometrica*, 22:265–90, 1954.

[2] Kenneth J. Arrow. An extension of the basic theorems of classical welfare economics. In J. Neyman, editor, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 507–532. University of California Press, Berkeley, 1951.

[3] B. D. Bernheim. Rationalizable strategic behavior. *Econometrica*, 52:1007–1028, 1984.

[4] J.Q. Cheng and M.P. Wellman. The WALRAS algorithm: A convergent distributed implementation of general equilibrium outcomes. Submitted for publication., 1995.

[5] D.R. Cheriton and K. Harty. A market approach to operating system memory allocation. In S.E. Clearwater, editor, *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, 1995.

[6] Vincent P. Crawford. Theory and experiment in the analysis of strategic interaction. Discussion Paper 95-37, Department of Economics, University of California, San Diego, September 1995.