Survivability through Market-Based Adaptivity: The MARX Project

Joseph E. Eggleston, Sugih Jamin, Terence P. Kelly, Jeffrey K. MacKie-Mason, William E. Walsh, and Michael P. Wellman[1]

University of Michigan
Ann Arbor, MI 48109 USA
{jeggle,sugih,tpkelly,jmm,wew,wellman}@umich.edu

Abstract

Market-based techniques represent a general approach to resource allocation in decentralized environments. In dynamic situations, the ability to find effective allocations of resources without central information and control is a crucial element of adaptivity. The Michigan Adaptive Resource eXchange (MARX) Project has developed market models for a variety of resource allocation problems--both canonical and application-specific. An overview of results from these efforts demonstrates the range of applicability of these methods, and their role in achieving survivable responses to resource shocks in distributed systems.

1. Market-Based Resource Allocation

A premise of inherent survivability is that a system can be made robust to *partially successful* attack through general architectural features. One way to achieve inherent survivability is through *adaptivity*: flexible response to unanticipated changes in environment. The key to successful adaptive behavior is *flexibility*—the ability to adapt to a range of adverse events without having to anticipate the particular responses in advance.

One important way to promote adaptivity is through *dynamic resource allocation*. Typically, the "unanticipated changes" that the system must respond to come in the form of lost or degraded resources, or new tasks that require resources. The response therefore takes the form of a new allocation of (remaining) resources to new and existing tasks. A flexible response through dynamic allocation is in principle possible in any context where resources may be redirected to multiple functions. That is, a resource originally intended for one use can be applied to another more critical use when the situation calls for it.

The problem then reduces to how to effectively perform dynamic resource allocation. For *inherent* survivability, the question is how to build this capability into the fundamental architecture of infrastructure for large-scale information systems.

For the past few years, the *MARX Project* (*Michigan Adaptive Resource eXchange*) has investigated a *market-based* approach: organize the system in terms of a *computational market,* where the entities (*agents*) controlling and capable of employing resources exchange them through a *market price system.* Building a computational market requires *commerce infrastructure*: basic services implementing negotiation, exchange (including payment in standard currencies), and other core functions of an operational economy.

There are many technical arguments—both computational and economic—in favor of adopting a market-based approach to dynamic resource allocation (Wellman and Wurman 1998a). We briefly mention a few of the more salient ones.

Decentralization. Large-scale information systems typically involve multiple, geographically and administratively distributed participants. No central source has the information or authority to make global decisions about how resources should be deployed. Markets are naturally decentralized according to agents representing distinct interests and capabilities.

Distributed decision and communication. Even when decisions could in principle be centralized, such an approach is often infeasible due to tractability, modularity, fault tolerance, or (most importantly in this context) vulnerability concerns. Markets are distributed in two fundamental ways. First, agents make local decisions based on their private information and objectives. Second, individual resources can be allocated according to separate mechanisms (i.e., each resource has its own price), with interdependencies among them accounted for by the agent's negotiation strategies.

Value-based allocation. Markets inherently account for the relative values and costs of resources and activities when determining allocations. This is in contrast to fixed priority schemes, common in most non-market systems. Agents indicate local values and costs in their negotiations, which are subsequently reflected in prices. Under certain conditions, markets are known to produce efficient (optimal) allocations despite the diversity of interests and locality of decision making.

[1] Authors alphabetical. This work was supported by DARPA/ITO grant F30602-97-1-0228. Project web site: http://ai.eecs.umich.edu/MARX

Embedding in economy. Information systems are not isolated, and indeed the activities mediated through the system ultimately connect to the real world. Thus, it is necessary (and desirable) to consider allocating resources not only within the system, but between entities in the system and the outside world. All real-world entities with significant use or production of resources are already equipped with a "market interface", that is, they naturally deal in economic terms, simply because markets are a "world standard".

This last reason suggests that market-based architectures are not only appropriate, but also in a sense inevitable. What may not be inevitable is an explicit fundamental connection between the external market economy and system infrastructure. We argue that such a seamless integration is in fact desirable, as imposing an additional, incompatible resource allocation mechanism for distinguished "system" resources introduces an arbitrary and artificial boundary. Indeed, the value of system resources ultimately lies in how they promote goals (or what their alternative uses are) outside the system, and so explicitly connecting system resource allocation to the external market economy seems more likely to lead to proper valuations.

Finally, the rapid development of infrastructure supporting all phases of electronic commerce will naturally serve as building blocks for development of market-based architectures for information systems. Thus, the market-based approach represents the ultimate "COTS" (commercial off-the-shelf) solution, both in terms of middleware services supporting market functions, and in terms of modules developed by commercial entities for market-based interaction within and outside the system boundaries.

2. The MARX Project

As indicated in the hierarchy of topics displayed in Figure 1, research in MARX has been divided into work on particular dynamic resource allocation problems, and development of generic infrastructure for deploying computational markets. Problems we have modeled in turn fall into two types: allocation of computational resources within information systems, and canonical allocation problems defined by problem structure rather than application. In order to assess the performance of pragmatic allocation mechanisms for these problems we have had to pursue methods for the assessment of *pragmatic* mechanisms, since the traditional theoretical analyses are limited to mechanisms that may provably not exist for our problems.

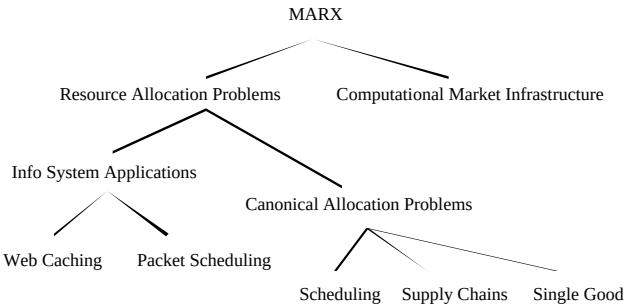

Figure 1: MARX project research topics.

2.1. Problems Studied

Only by examining specific information-system applications can we compare results from the computational market with alternative approaches, and only by generalizing to domain-independent problem classes can we establish that our methodology will cover a broad range of survivability contexts.

Under information system applications, we have chosen two important and very well-studied problems—web caching and packet scheduling. These two examples stand for large classes of important problems. The first is an important instance of distributed data *storage*; the second an instance of data *transport*. As described below, these problems admit multiple market-based approaches, which in some instances can be related to well-known non-market allocation schemes. This facilitates evaluation of allocation quality as well as adaptivity properties.

Our canonical allocation problems are designed to represent important patterns of resource allocation that occur across a multitude of application contexts. The simplest are the "single good" problems, which concern the negotiation of transfers of a single resource type. The simple structure of these problems makes them amenable to in-depth analysis, including a near exhaustive consideration of negotiation mechanisms and strategies. Understanding the single-good problem (which is also very well characterized in the economic literature) is a prerequisite to informed design of computational markets for more complex problems involving multiple resource types.

Problems involving complex interrelated activities invariably involve multiple resource types. Although these resources can be allocated through multiple single-resource mechanisms, the interdependencies among the resources significantly complicate the problem of effective resource allocation. In our MARX research, we have been exploring two particularly common and challenging source of interdependencies. First, in *scheduling* problems, resources have temporal dependencies, meaning that the value of a resource for a particular use depends fundamentally on *when* it is employed (e.g., whether it enables the user to meet a deadline). We have some particular protocols for market based scheduling. The techniques work well for some problem classes, but

others can be fundamentally difficult to decentralize (Walsh et al. 1998; Wellman et al. to appear). Second, in *supply chain* problems (Walsh and Wellman 1999b), dependencies are prerequisite relationships between elements in the chain. We have captured an interesting class of supply chain problems in our task dependency network model described below. In addition, we have considered combinations of scheduling and supply chain problems, involving both sorts of dependencies. We believe that these two patterns represent the important dependencies arising in a large class of allocation problems underlying large-scale systems. These resource allocation problems serve as the source of dynamic allocation scenarios that can demonstrate inherent survivability through market-based adaptivity. Specific applications represent potential domains for the demonstrations, whereas the canonical problems dictate our approach to mapping our techniques to new domains. The computational infrastructure for market-based allocation we have developed in the MARX project provides a vehicle for deploying these demonstrations.

2.2. Methods

In our research we are developing methods for *pragmatic* mechanism design. A mechanism is a set of rules that specifies allocations as a function of messages from the agents who hold private information (an auction is a classic example). There is an extensive theoretical literature on the design of resource allocation mechanisms (Campbell 1987; Mas-Colell, Whinston, and Green 1995). The standard approach is to seek a mechanism that always yields an allocation that is maximally efficient subject to plausibility constraints on agent behavior. A typical set of constraints is that agents are rational in the sense of playing Bayesian-Nash strategies; their messages are consistent with self-interest (and thus may not be truthful revelations of their private information); and they (including the mediator or auctioneer) will not participate unless they expect to be no worse off as a consequence of participating. Unfortunately in the *negotiation* problems on which we focus, well-known results establish that *no* mechanism (market or otherwise) exists that satisfies this set of seemingly reasonable constraints while always guaranteeing efficient allocations (Gibbard 1973; Myerson and Satterthwaite 1983). Obviously, for the reasons above and others, market-based mechanisms are still of considerable real world importance. We thus have been developing methods to design and assess *feasible* mechanisms that cannot satisfy the full set of standard desiderata.

One of our methods is conventional theoretical performance analysis, but subject to the *feasibility* constraint: we relax one of the assumptions on plausible behavior enough to avoid the impossibility results cited above. For example, in our study of market mechanisms for scheduling problems, we have sharp results on performance when agents are restricted to natural but not fully rational bidding strategies (Wellman et al. to appear). Another method is to simulate allocation outcomes under reasonable assumptions on agent behavior. We have used this method to assess allocation efficiency (Anderson, Birgean, and MacKie-Mason 1999; Wurman 1999)(Kelly et al. 1999a, 1999b; Callaway or other packet scheduling work). We have used both of these methods to assess the performance of a non-price economic allocation mechanism (the Generalized Vickrey Auction (MacKie-Mason and Varian 1994)). For example, in our application of the GVA to scheduling (Wellman et al. to appear), we relax the requirement that the auctioneer balance its budget. In a single-good context (Anderson, Birgean, and MacKie-Mason 1999), we give up some efficiency in order to balance the budget through participation and transaction fees.

3. Information System Applications

Our argument that dynamic resource allocation via computational markets yields inherent survivability hinges on the premise that most important applications can be cast in this framework. We have significant experience in mapping a wide range of problems to the market framework, including those described here and others (Wellman 1996). Most practical resource allocation problems fall in one of the four categories formed by crossing discrete or continuous quantity scales with discrete or continuous time scales. Therefore, many interesting problems fit models we have already designed and tested. In this section we describe two particular applications, to resource allocation problems arising in networked information systems.

3.1. Wide-Area Storage Allocation (Web Caching)

The problem of *Web caching* is that of selectively allocating storage on a wide-area network in such a way as to maximize value to system users. Shared Web caches provide different benefits for different classes of users: latency reduction for clients, congestion reduction for all network users, and load reduction for servers. Disk space in shared Web caches is a strategically-placed scarce resource that may be diverted to serve some users at the expense of others, and therefore these caches are ideal loci for differential quality-of-service (QoS) mechanisms. We have explored two kinds of mechanisms that bias the allocation of cache space toward system users (servers and clients) who value caching most. One approach is to generalize conventional cache replacement policies such as the “least frequently used” (LFU) algorithm. Another is to design explicit market allocation mechanisms in which agents bid for disk space in shared caches. We have pursued both of these methodologies and have evaluated several designs in comparison with conventional cache management algorithms (Williams et al. 1996).

It is sometimes possible to establish tight bounds on the performance of cache replacement policies independent of workload (client document request patterns (Irani 1997;

Sleator and Tarjan 1985). This approach, however, yields performance guarantees far weaker and more pessimistic than the observed performance of most reasonable removal policies under real workloads. In order to evaluate the relative performance of practical caching schemes we must therefore employ *trace-driven simulation*, in which a cache simulator processes workloads logged at actual caches. The National Laboratory for Applied Network Research (NLANR) currently operates ten large Web caches nationwide, each of which serves many corporate- and campus-sized client networks. We obtained client request data collected at six of these caches over a four-week period and wrote a general-purpose cache simulator to support trace-driven experiments. A distinguishing feature of our evaluation methodology is that our main performance metric is *value delivered to system users* rather than standard metrics such as byte hit rate (fraction of requested data served from cache rather than servers). By diverting disk space to serve the needs of those who most value caching (variable QoS), we aim to increase the aggregate value of the cache (utility maximization), as compared with value-insensitive replacement policies.

In the LFU replacement policy, unpopular data are flushed from the cache in favor of heavily requested data. Formally, if n_u is the number of requests for URL u and $size_u$ is its size, an LFU cache stores URLs with the highest observed values of n_u. Let $size_u V_u$ be the benefit that the server associated with u receives when requests for u are served from cache rather than by the server. Our *weighted LFU* cache replacement policy stores URLs with the highest value of $n_u V_u$, because these are precisely the items that generate the most aggregate utility per unit cache size. The potential value generated by URL u per unit size is given by

$$\sum_{requests\ for\ u} (size_u V_u / size_u),$$

which is simply $n_u V_u$.

Weighted LFU is interesting because it corresponds exactly to the policy resulting from a market-based mechanism: it is natural to interpret $size_u V_u$ as a server's *payment* to a profit-maximizing cache each time the cache serves u. Experiments have shown that this simple generalization of LFU delivers substantially higher aggregate value to servers than ordinary LFU for some distributions of V_u. For example, Figure 2 presents simulation results for a two-week interval of NLANR data for a large cache site (Kelly et al. 1999). Our performance metric, *value hit rate*, measures the fraction of request value delivered to users, where value is given by the weights V_u. As the graph shows, weighted LFU outperforms LFU or LRU (the *least-recently-used* replacement policy) for any given cache size. Weighted LFU performs even better when augmented with a tunable aging mechanism that prevents the cache from becoming cluttered with formerly-popular documents that are no longer requested (Kelly, Jamin, and MacKie-Mason 1999). Extensive experiments have shown that weighted LFU with aging usually delivers more aggregate value to servers than "best-of-breed" algorithms from the Web caching literature, e.g., "Greedy-Dual Size" (Cao and Irani 1997).

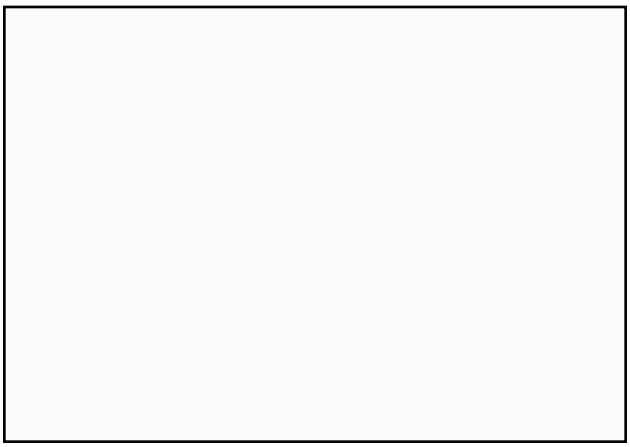

Figure 2: In this trial, weighted LFU (wLFU) provided higher value than unweighted frequency- or recency-based policies (uLFU and pLRU).

Whereas weighted LFU successfully *exploits* server valuation information to deliver high value to system users, it is not likely to *obtain* this private information: servers would generally find it advantageous to report false V_u values. An *incentive-compatible* allocation mechanism has the property that truthful revelation of valuation information is a dominant strategy for participating agents. In our implementation, weighted LFU is not incentive compatible. By contrast, a periodic auction market in which servers and clients directly bid for cache disk space *can* be incentive compatible, given the right choice of auction. We have investigated a scenario in which servers with simple and reasonable bidding strategies contend for disk space via periodic incentive-compatible auctions. We characterize some conditions under which this particular system can obtain higher overall value than even weighted LFU with aging (Chan et al. 1999).

3.2. Packet Scheduling

The fundamental feature of a market-based packet scheduling system is that it provides differential QoS to packets based on willingness to pay, as expressed in *bids*. We achieve this through a bid-sensitive *queuing system*, consisting of two parts: the *scheduling discipline* which determines the next packet to be served, and a *dropping policy* which determines the packet to drop when the queue is full.

3.2.1. Scheduling Disciplines

We study two scheduling disciplines that can provide market-based QoS: Lottery Scheduling (Waldspurger and Weihl 1994) and Deterministic (Static Priority) Scheduling. For comparison purposes, we also include traditional FIFO (first-in-first-out) scheduling in our study.

Lottery Scheduling is a probabilistic method, originally proposed for CPU scheduling. The next packet to be served is chosen by holding a lottery, with the probability of an individual packet being served being proportional to its bid value:

$$\Pr(\text{packet } k \text{ is served}) = B_k / \sum_j B_j,$$

where B_i is the bid value of packet i and j ranges over the packets in queue.

With lottery scheduling, packets with higher bids have a greater chance of being served next. Therefore, during times of congestion higher bid packets should typically experience lower queuing delay than packets with lower bids. However, since low bid packets nevertheless will have a chance of being served, they will not be starved. Even if high bid packets keep arriving, low bid packets will still receive a share of the bandwidth proportional to their bid value.

One side effect of Lottery Scheduling is that packets in a flow have a higher probability of arriving out of order at the receiver. When a flow has more than one packet in a router's queue, each packet has equal probability of being forwarded next (assuming the flow has not changed the bid value carried by its packets). In order to ensure proper ordering, we would have to maintain an additional queue *for each flow*.

Deterministic Scheduling always forwards the packet with the highest bid. If the queue contains multiple packets with the same highest bid value, they are served in FIFO order. As long as a flow does not change its bid value, Deterministic Scheduling will not reorder queued packets belonging to a flow.[2] The disadvantage of Deterministic Scheduling is the possibility of starvation; if higher bid packets keep arriving, lower bid packets will never be served.

3.2.2. Dropping Policies

We consider three dropping policies: Lottery Drop, Deterministic Drop, and the standard Drop Tail policies.

The mechanism behind Lottery Drop is analogous to that of Lottery Scheduling. When a new packet arrives at a full queue it is placed in the queue and a lottery is held to determine which packet to drop. For Lottery Drop the probability of an individual packet being dropped is proportional to the inverse of its bid value.

$$\Pr(\text{packet } k \text{ is dropped}) = (1/B_k) / \sum_j (1/B_j).$$

Hence higher bid packets are less likely to be dropped than lower bid packets.

Deterministic Drop selects the lowest bid packet in the queue to drop. If there is more than one packet with the same lowest bid, the latest to arrive is dropped.

3.2.3. Interactions of Scheduling and Dropping Policies

As stated above, a queuing mechanism is defined by its scheduling discipline and dropping policy. When combined into a single queuing mechanism, the scheduling discipline and dropping policy are no longer independent. For example, focusing on the time spent by a single packet in the queue, the number of packets served prior to this packet influences the chances of this packet being dropped. Likewise, the number of other packets dropped influences the amount of time the packet must wait before it sees service. Hence, it is important to compare complete queuing mechanisms rather than just the forwarding or dropping policies. The following are the combinations of scheduling disciplines and dropping policies we have studied:

1. FIFO scheduling with Drop Tail (FSDT). This is the traditional router queuing mechanism.
2. Lottery Scheduling with Drop Tail (LSDT). Lottery Scheduling provides lower latencies for higher bid packets. Drop Tail guarantees that once a packet enters the queue it will eventually be served.
3. Lottery Scheduling with Lottery Drop (LSLD). This combination tends to favor higher bid packets in both forwarding and dropping. A packet that has entered the queue is not guaranteed to be forwarded.
4. FIFO Scheduling with Lottery Drop (FSLD). This combination favors higher bid packets only when there is enough congestion to cause the queue to overflow.
5. Deterministic Scheduling with Deterministic Drop (DSDD). This combination always forwards the highest bid packet and drops the lowest bid packet.

3.2.4. Experiments

We have investigated the relative ability of the various queuing mechanisms to differentiate QoS using an experimental testbed implemented in the FreeBSD 3.2 operating system kernel. Results on constant flow priorities (to be reported elsewhere) confirm the value-sensitivity of lottery-based schemes compared to FIFO and drop tail, and their flexibility relative to deterministic schemes. Initial results on adaptive behavior with dynamic flow priorities are reported in Section 4.2 below.

4. Dynamic Allocation Scenarios

We consider two kinds of dynamic allocation scenarios to demonstrate the inherent survivability of market-based architectures. In both, a "normally operating" system is subjected to a resource shock—a sudden variation in resource availability or need—and must dynamically reallocate to address the qualitative change in environment. In the first, which we call a *discrete* scenario, the system has achieved a steady-state allocation before the shock event, which may consist of a lost asset, loss of a key participant, or insertion of a new high priority task. In the second, *continuous* scenario category, the system never reaches a steady state, as some amount of continual variation of resources and tasks is always present. The resource shock in this case is a qualitative jump in the degree of this variation.

[2] However, packets can still arrive out of order at the receiver due to network topological or routing changes.

Dynamic properties necessary for survivability can be exhibited in the context of both kinds of scenarios. We have developed in-depth models of instances of both discrete and continuous allocation models. In this section we describe abstract versions of discrete and continuous resource-shock scenarios, in turn.

4.1. Discrete Shocks in Task Dependency Networks

In complex activities, achievement of an overall objective may require accomplishment of various tasks, which themselves may require subtasks, and so on. A plan will therefore specify an entire network of task accomplishment, with each path in the network representing a *supply chain*, that is, a sequence of relationships where one task is provided as input to the next in the chain, until the final objective is achieved.

For example, consider a hypersimplified military scenario, where a commander wishes to execute an air attack. Air attacks require a bomber squad and a fighter squad, and these in turn require airfields.[3] Given a specification of the available resources and task-achievement options, we can formulate a *task dependency network* (Walsh and Wellman 1998) describing the possible configurations. A *solution* to the planning problem is a subnetwork specifying the assignment of tasks and subtasks so that the end objective is satisfied. When there is no resource contention, as in the network shown in Figure 3, deriving an allocation supporting the air attack is simple.

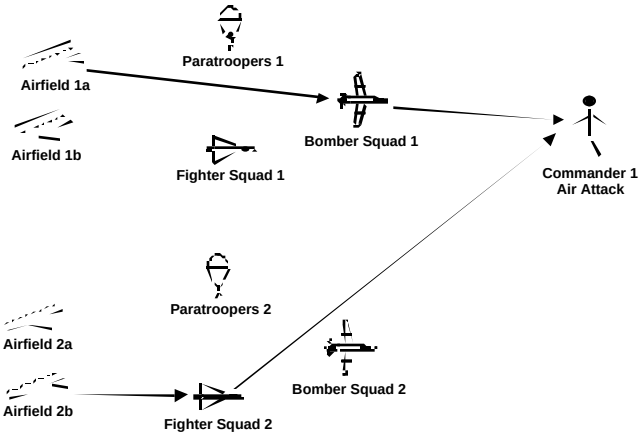

Figure 3: A task dependency network, no resource contention.

[3] Squads and airfields might be viewed as resources rather than tasks, unless we interpret the squad, for example, as just a shorthand for the mission the squad performs. Which view is more natural is situation-dependent, and so we use the terms task and resource interchangeably in this discussion. Note that in this example, we assume that the different forces are interchangeable, within each type (e.g., bomber squads 1 and 2 are equivalent with respect to what they can accomplish).

If a new task arrives, however, as represented by "Commander 2" in Figure 4, resource requirement may conflict with the previous allocation. In the example, the ground attack cannot be accomplished, because there are not enough airfields to support the units required for both attacks.

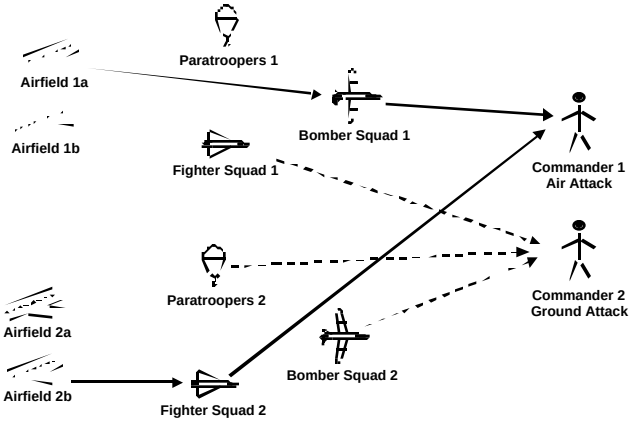

Figure 4: A second task objective introduces resource contention.

If the new task is higher priority than the old, then we would like the system to adapt by dynamically reallocating the airfields to squads participating in the new activity. A market-based system can perform this reallocation as follows. The diagram of Figure 5 depicts a computational market in tasks and resources, with agents rendered as rectangles and exchangeable goods as circles. The shaded elements represent the steady-state solution to the initial situation, without resource contention (prices not shown).

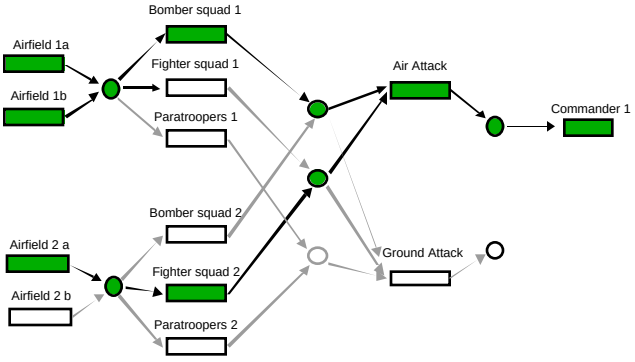

Figure 5: A computational market representing the task dependency network.

To represent the resource shock, we introduce a new agent, Commander 2, who places a significantly higher value on its mission than did the first commander. The market, starting from the previous allocation and prices, adjusts prices until no agents change their bids. In the new steady-state solution, the higher priority task is achieved, and the lower priority task is unaffordable. (If both were possible, then both could be supported by the

market.) The extended network, with task valuations and final prices, is shown in Figure 6.

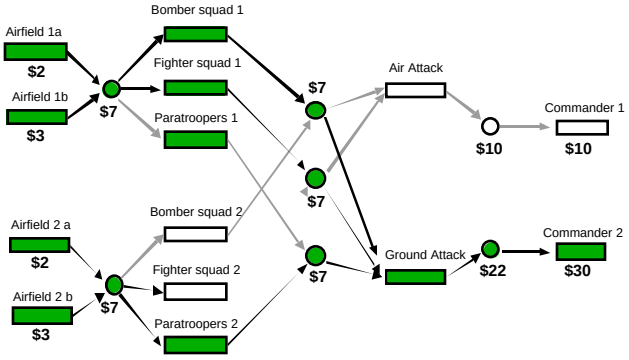

Figure 6: Allocation and prices on introduction of Commander 2's task.

In this case, contention for the scarce airfields drives their price to $7, even though airfield "suppliers" have costs of only $2 or $3. The force producers sell their services for $7, and thus make no profits. The ground attack producer makes a slight profit, selling its service for $1 more than the combined cost of its force inputs. Note that the inactive (unshaded) producers cannot make a profit at the given prices, and so drop out. Although the specific prices resulting depend on the agent strategies and order of communication, the general pattern where profitable producers are active and unprofitable ones not so is a general characteristic of these protocols.

We have developed a formal characterization of these task dependency networks, and have studied a precise market protocol, specifying the allocation mechanisms and agent bidding strategies (Walsh and Wellman 1998). We have found that a broad class of networks reliably converge to a solution—even in a completely asynchronous setting—as long as the value of the end task sufficiently exceeds its ultimate cost (Walsh and Wellman 1999a). This implies that the protocol successfully adapts to discrete resource shocks of the sort described above, if the new task has sufficiently higher value than the others in the network. Similarly, the protocol can also handle necessary reallocations in case of loss of resources, loss of agents, or exogenous increases in cost of resources, so long as some tasks have sufficiently high value relative to the costs of surviving resources.

In the present discussion, we assumed that we could continue the market at the previous prices in response to a shock. While this is a realistic assumption if the shock occurs during negotiation, agents may not wish to continue as such if the exchange contracts have been finalized. We consider it an interesting avenue for future work to analyze protocols for agents to reallocate tasks for which they have acquired rights.

The task dependency network model is quite general, covering a broad range of planning and scheduling problems involving limited resources. Formally, the problem is NP-hard, which means that any propositional satisfiability (SAT) problem can be encoded in a task dependency network. As a result, the market protocol implements a decentralized procedure for a broad range of combinatorial optimization problems.[4]

4.2. Shocks in a Continuous Network Allocation Problem

Unlike the episodic task allocation considered above, in *continuous* dynamic scenarios, the commodities themselves are typically controllable at fine grain. In addition, the allocation is continuously updated over time as computational and communicational activities originate and terminate on the system. Resource shocks take the form of sudden changes in the profile of resources available or activities demanded.[5]

In preliminary studies of the packet scheduling problem (Section 3.2), we have verified some expected qualitative behaviors of adaptation to continuous resource shocks. Upon shocks (such as a sudden increase in priorities for a fraction of service requests), a system under the control of lottery scheduling rapidly transitions to an appropriate revised profile of service levels. The speed and sharpness of the transition depends significantly on the volume and nature of the fraction of traffic that is unchanged by the shock.

For example, Figure 7 shows the response of network traffic to two shocks, measured in terms of the percentage of sent packets received (i.e. not dropped due to buffer overflow). We initially experiment with 40 connections all submitting the same bid value with each packet to be delivered. Around 100 seconds into the experiment, half of the connections increase their priority (and hence bid values) five-fold. As the graph shows, when network switches implement the lottery drop (LD) policy, connections that increase their bid values immediately see improvements on the percentage of packets received at the destinations. This comes at the cost of worse service experienced by lower bidding connections. At around 200 seconds into the experiment, half of the connections that earlier raised their bid value double them again. These connections again immediately see improvements in service over those that do not raise their bids. Finally, at around 300 seconds into the experiment, the highest bidding connections lower their bids back down by half. This results in half of the connections again receiving the same service that is better than the service seen by the other half that never raised its bid value. Figure 7 also shows that if in addition to lottery drop network switches

[4] We are not claiming that the particular decentralization arising from the SAT reduction is a natural one or useful per se. However, it is perhaps surprising that such problems can be solved reliably without systematic or explicitly stochastic search.

[5] In general the quantity and time scales need not match: there are common problems involving continuously varying quantities allocated at discrete intervals; likewise there are interesting problems in the allocation of discrete goods continuously over time (e.g., airport landing slots, (Rassenti, Smith, and Bulfin 1982)).

also implement lottery scheduling (LS), as opposed to first-in-first-out scheduling (FS), packets with higher bids see additional preferential treatment.

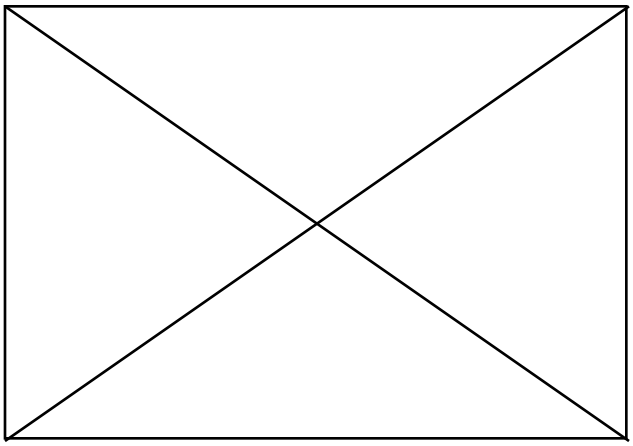

Figure 7: Response to shock with constant bit rate transmissions.

All the connections in the experiment described above transmitted at constant bit rate. Researchers in network traffic characterization have observed long-range dependency in aggregate network traffic (Leland et al. 1994). To study the effectiveness of our market-based packet scheduling mechanism on long-range dependent traffic, we conduct a similar experiment on sources generating on-off traffic with Pareto distributed on and off times. Aggregate traffic from such sources has been shown to exhibit long-range dependency (Willinger 1995). Figure 8 shows that in the face of long-range dependent traffic, while higher bidders continue to receive preferential treatment under lottery drop, and an exaggerated preferential treatment under lottery scheduling, lower bidders do not suffer as much as in the previous case. The high variance of long-range dependent traffic allows lower bidding traffic to continue to be served at network switches, albeit with a longer delay. Hence, when network traffic is very bursty, lower bidding traffic experiences longer queuing delay but not higher loss rate.

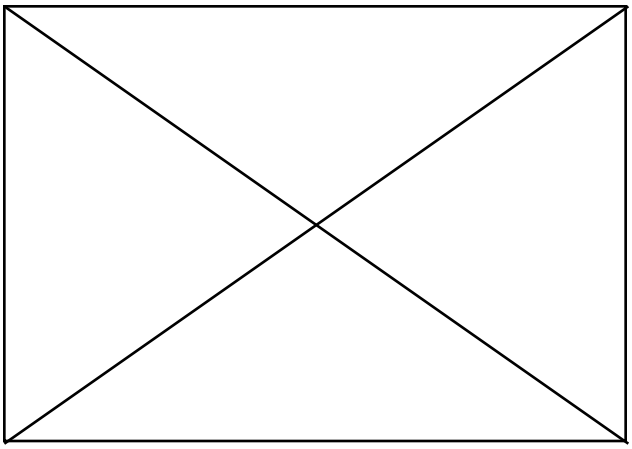

Figure 8: Shock response with Pareto-distributed background traffic.

5. Computational Market Testbed

Our computational market infrastructure is based on the Michigan Internet AuctionBot,[6] a configurable auction server that supports a wide range of negotiation mechanisms. The AuctionBot has a web interface for human traders, as well as an API for software agents. In addition, we have built extended modules ("agentware") to facilitate development of software traders, and tools for specification of entire market configurations. Thus, the AuctionBot serves as a general tool for rapid construction of market-based demos.

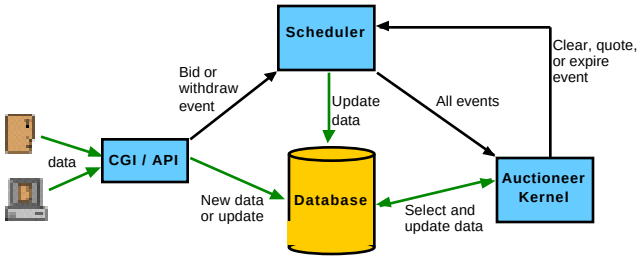

Figure 9: AuctionBot architecture and data flow diagram.

The basic organization and operation of the AuctionBot is depicted in Figure 9. Human or software agents create auctions or submit bids via web or programming interfaces (O'Malley and Kelly 1998). A scheduler maintains a queue of pending events—processing bids, clearing auctions, or releasing information. An event is activated when it rises to the top of the queue (i.e., all necessarily prior events have been initiated) *and* its earliest activation time (if any) is past. The schedule passes activated events to the respective auctioneer kernels, which ensure that the events are executed in a manner respecting temporal consistency

[6] http://auction.eecs.umich.edu/

(Wellman and Wurman 1998b). A relational database maintains persistent data about bids, agents, and auction specifications, providing availability of relevant negotiation information to agents even when auction processes are inactive or excessively busy.

The AuctionBot is designed to cover a large class of conceivable auction mechanisms, as defined by our systematic parametrization of the design space (Wurman, Wellman, and Walsh to appear). This configurability is implemented by a flexible multi-level scheme supporting multiple representations for bids and order books (sets of bids), and multiple clearing algorithms implementing allocation rules. As depicted in Figure 10, some characteristics of bid and order-book representations are generic to all auctions, some depend on the form of bids allowed, and yet others are optimized for particular clearing algorithms. For example, the "4-heap" data structure supports a broad class of allocation mechanisms for discrete goods (Wurman, Walsh, and Wellman 1998), but can be overridden in a modular manner to support alternative or new mechanisms.

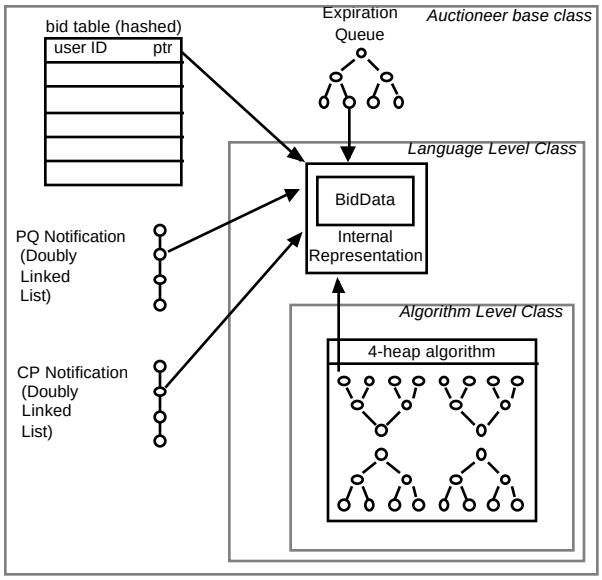

Figure 10: Flexible multi-level data structures for bids and order books.

Current work is developing AuctionBot functionality further, in particular (1) augmenting the range of negotiation mechanisms, (2) improving tools for specification of scenarios, (3) and extending the API and agentware, especially to facilitate participation by third-party agent developers.[7]

Other technical enhancements to the market protocols underway are designed to improve their acceptability in real distributed computation environments. These include

- Protocol layers addressing security and privacy requirements (Harkavy, Tygar, and Kikuchi 1998; Kelly 1998; Naor, Pinkas, and Sumner 1999).
- A protocol layer for distributed detection of quiescence in the negotiation process (e.g., when a steady state is reached after a shock). Our approach (Wellman and Walsh 1999) is based on an extension of the Dijkstra-Scholten algorithm for detecting termination in a diffusing computation (Dijkstra and Scholten 1980; Lynch 1996).

6. Discussion: Evaluating Adaptive Behavior

To make statements about inherent survivability based on our work, we require evaluation criteria for the adaptivity properties of our demonstrated architecture.[8]

Adaptation is required only in the event of unanticipated change, so we define adaptation scenarios in terms of the cause and nature of changes. Adverse changes in fundamental operating conditions ("failures") may be caused by intentional attacks as well as accidental causes (e.g., hardware or software bugs). Other changes in the environment may not be considered failures, but may nevertheless entail significant adaptation due to new directives, loss of resources, additional resources introduced, or other revisions in capabilities. For our purposes the cause of the resource shock is typically not relevant, as the allocation problem is the same regardless of whether the change in resource environment is due to an intentional attack or accidental event.[9]

The second question is how to evaluate an adaptation episode. The standard of performance is with respect to the behaviors possible *given* the resource shock; it is not useful to compare performance to what could have been achieved in the absence of a shock. We presume that the system is comprised of multiple participants (i.e., we have a distributed or multiagent system), each with its own objectives or preferences over possible *outcomes*. These outcomes in general characterizes complete temporal courses of activity, not just instantaneous states. The overall *welfare* of a system is some function of the outcomes of all of the respective participants, accumulated over time. For example, one simple welfare function would be a time-discounted sum of utility functions

[7] The AuctionBot has been operational and available over the Internet since 1997, and versions of our agentware are already available.

[8] Some of this discussion is based on the report of the "Measuring Adaptivity" working group at the DARPA/ITO Workshop on Adaptive Architectures for Information Survivability, May 1998. Group members were V. Lesser, F. Webber, M. Wellman, D. Wells, and Y. Yemini. Views expressed here, of course, are our sole responsibility.

[9] However, the cause is highly relevant to anticipation and prevention of future resource shocks. We view this as outside the scope of what market-based architectures provide. Prediction and prevention techniques (e.g., intrusion detection) are thus complementary rather than competitive methods.

representing the respective preferences of all of the participants.

For example, consider the response of the task network protocol to discrete shocks, as discussed in 4.1. We have been able to establish that under certain conditions, the protocol responds successfully to adverse shocks, such as the loss of a basic resource or of some production capability, or increase in cost or degradation of same. "Successfully" here means that if a good enough solution exists in the degraded state, then the protocol will find it when resumed from a previous steady state.

This characterization of adaptivity is quite helpful, but it does not provide a quantitative measure of the *quality* of solutions reached. We have begun to evaluate the quality of solutions found for the baseline market-based task network protocol (without shocks), by measuring the *total surplus* achieved by all agents in the system (Walsh and Wellman 1999a). The surplus for a given agent is the difference between the value of what it obtains, and what it pays. In a series of randomly generated task allocation networks, we found that solutions achieved about 83% of the optimal surplus on average. Most of the efficiency loss was due to agents that purchased one or more inputs without selling their outputs. By extending these protocols to permit *decommitment* (Sandholm and Lesser 1996) in such cases, we can recover much of the inefficiency, and achieve 97% of total surplus on average (Walsh and Wellman 1999a).

These results are merely suggestive, as exact values will depend on the method for generating problem instances, among other factors. Moreover, for evaluating adaptivity we must examine specific shock scenarios. In this case, we would expect the system to achieve comparable fractions of available surplus for the case of shocks attributed to lost resources, rising resource costs, or the introduction of higher valued tasks. Whether the protocol would prove to be as effective at adapting to the availability of new resources or production capabilities is an open question.

In related work, we have emphasized the importance of evaluating the time path of performance, and comparing different solutions based on their time-discounted sum of agent utilities over time (Brooks et al. 1999). For example, one solution might be guaranteed to yield the highest value once the adjustment to the new situation is complete, whereas another has a lower ultimate value, but rises towards its steady-state value much more quickly. Enough advantage over the earlier periods following the shock may outweigh, after discounting, permanent long run gains from the former approach.

Finally, in evaluating adaptivity we must consider some overarching criteria that go beyond quality of allocations achieved. These include online costs, such as overhead of the adaptive infrastructure, as well as offline costs, such as the effort in building the infrastructure, creating interfaces for system components, and modeling domains so they can effectively exploit the adaptive features of the system.

References

Anderson, Axel, Ionel Birgean, and Jeffrey K. MacKie-Mason. 1999. Bilateral negotiation with fees. In *IBM/IAC Workshop on Internet-Based Negotiation Technology*, Yorktown Heights, NY.

Brooks, Christopher H., Scott Fay, Rajarshi Das, Jeffrey K. MacKie-Mason, Jeffrey O. Kephart, and Edmund Durfee. 1999. Automated Stategy Seaches in an Electronic Goods Market: Learning and Complex Price Schedules. In *ACM Conference on Electronic Commerce*, Denver.

Campbell, Donald E. 1987. *Resource Allocation Mechanisms*: Cambridge University Press.

Cao, Pei, and Sandy Irani. 1997. Cost-aware WWW proxy caching algorithms. In *USENIX Symposium on Internet Technologies and Systems.*

Chan, Yee Man, Jonathan Womer, Jeffrey K. MacKie-Mason, and Sugih Jamin. 1999. One size doesn't fit all: Improving network QoS through preference-driven Web caching. In *Second Berlin Internet Economics Workshop.*

Dijkstra, Edsger W., and C. S. Scholten. 1980. Termination detection for diffusing computations. *Information Processing Letters* **11**:1-4.

Gibbard, Allan. 1973. Manipulation of voting schemes: A general result. *Econometrica* **41**:587-601.

Harkavy, Michael, J. D. Tygar, and Hiroaki Kikuchi. 1998. Electronic auctions with private bids. In *Third USENIX Workshop on Electronic Commerce*, Boston.

Irani, Sandy. 1997. Page replacement with multi-size pages and applications to Web caching. In *Twenty-Ninth ACM Symposium on the Theory of Computing*, El Paso, TX.

Kelly, Terence P. 1998. Secure anonymous auctions without trusted parties: University of Michigan.

Kelly, Terence P., Yee-Man Chan, Sugih Jamin, and Jeffrey K. MacKie-Mason. 1999. Biased replacement policies for web caches: Differential quality-of-service and aggregate user value. In *Fourth International Web Caching Workshop*.

Kelly, Terence P., Sugih Jamin, and Jeffrey K. MacKie-Mason. 1999. Variable QoS from Shared Web Caches: User-Centered Design and Value-Sensitive Replacement.

Leland, W. E., M. S. Taqqu, W. Willinger, and D. V. Wilson. 1994. On the Self-Similar Nature of Ethernet Traffic (Extended Version). *IEEE Transactions on Networking* **2**:1-15.

Lynch, Nancy A. 1996. *Distributed Algorithms*: Morgan Kaufmann.

MacKie-Mason, Jeffrey K., and Hal R. Varian. 1994. Generalized Vickrey Auctions: University of Michigan.

Mas-Colell, Andreu, Michael D. Whinston, and Jerry R. Green. 1995. *Microeconomic Theory*: Oxford University Press.

Myerson, Roger B., and Mark A. Satterthwaite. 1983. Efficient mechanisms for bilateral trading. *Journal of Economic Theory* **29**:265-281.

Naor, Moni, Benny Pinkas, and Reuben Sumner. 1999. Privacy Preserving Auctions and Mechanism Design. In *ACM Conference on Electronic Commerce*, Denver.

O'Malley, Kevin, and Terence Kelly. 1998. An API for Internet auctions. *Dr. Dobb's Journal* (September):70-74.

Rassenti, S. J., V. L. Smith, and R. L. Bulfin. 1982. A combinatorial auction mechanism for airport time slot allocation. *Bell Journal of Economics* **13**:402-417.

Sandholm, Tuomas W., and Victor R. Lesser. 1996. Advantages of a leveled commitment contracting protocol. In *Thirteenth National Conference on Artificial Intelligence*, Portland, OR.

Sleator, Daniel, and Robert Tarjan. 1985. Amortized efficiency of list update and paging rules. *Communications of the ACM* **28**:202-208.

Waldspurger, Carl A., and William E. Weihl. 1994. Lottery scheduling: Flexible proportional-share resource management. In *Proceedings of the First Symposium on Operating System Design and Implementation (OSDI)*.

Walsh, William E., and Michael P. Wellman. 1998. A market protocol for distributed task allocation. In *Third International Conference on Multiagent Systems*, July, Paris.

Walsh, William E., and Michael P. Wellman. 1999a. Efficiency and equilibrium in task allocation economies with hierarchical dependencies. In *Sixteenth International Joint Conference on Artificial Intelligence*, Stockholm.

Walsh, William E., and Michael P. Wellman. 1999b. Modeling supply chain formation in multiagent systems. In *IJCAI-99 Workshop on Agent-Mediated Electronic Commerce*, Stockholm.

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

Wellman, Michael P. 1996. Market-oriented programming: Some early lessons. In *Market-Based Control: A Paradigm for Distributed Resource Allocation*, edited by S. Clearwater: World Scientific.

Wellman, Michael P., and William E. Walsh. 1999. Distributed quiescence detection in multiagent negotiation. In *AAAI-99 Workshop on Negotiation*, Orlando, FL.

Wellman, Michael P., William E. Walsh, Peter R. Wurman, and Jeffrey K. MacKie-Mason. to appear. Auction protocols for decentralized scheduling. *Games and Economic Behavior.*

Wellman, Michael P., and Peter R. Wurman. 1998a. Market-aware agents for a multiagent world. *Robotics and Autonomous Systems* **24**:115-125.

Wellman, Michael P., and Peter R. Wurman. 1998b. Real time issues for Internet auctions. In *IEEE Workshop on Dependable and Real-Time E-Commerce Systems*, June, Denver, CO.

Williams, Stephen, Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, and Edward A. Fox. 1996. Removal policies for World-Wide Web documents. In *ACM SIGCOMM.*

Willinger, W. and Taqqu, M.S. and Sherman, R. and Wilson, D.V. 1995. Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level. In *ACM SIGCOMM.*

Wurman, Peter R. 1999. Market Structure and Multidimensional Auction Design for Computational Economies. PhD, Computer Science and Engineering, University of Michigan.

Wurman, Peter R., William E. Walsh, and Michael P. Wellman. 1998. Flexible double auctions for electronic commerce: Theory and implementation. *Decision Support Systems* **24**:17-27.

Wurman, Peter R., Michael P. Wellman, and William E. Walsh. to appear. A parametrization of the auction design space. *Games and Economic Behavior.*