Middle-Agents for the Internet*

Keith Decker
Computer and Information Sciences
University of Delaware
decker@cis.udel.edu

Katia Sycara and **Mike Williamson**
The Robotics Institute
Carnegie-Mellon University
(sycara,mikew)@cs.cmu.edu

Abstract

Like middle-men in physical commerce, middle-agents support the flow of information in electronic commerce, assisting in locating and connecting the ultimate information provider with the ultimate information requester. Many different types of middle-agents will be useful in realistic, large, distributed, open multi-agent problem solving systems. These include *matchmakers* or *yellow page* agents that process advertisements, *blackboard* agents that collect requests, and *brokers* that process both. The behaviors of each type of middle-agent have certain performance characteristics—privacy, robustness, and adaptiveness qualities—that are related to characteristics of the external environment and of the agents themselves. For example, while brokered systems are more vulnerable to certain failures, they are also able to cope more quickly with a rapidly fluctuating agent workforce and meet certain privacy considerations. This paper identifies a spectrum of middle-agents, characterizes the behavior of three different types, and reports on initial experiments that focus on evaluating performance tradeoffs between matchmaking and brokering middle-agents, according to criteria such as load balancing, robustness, dynamic preferences or capabilities, and privacy.

1 Introduction

One of the basic problems facing designers of open, multi-agent systems for the Internet is the connection problem [Davis and Smith, 1983]—finding the other agents who might have the information or other capabilities that you need. There are two special types of information used in this process—*preferences* and *capabilities*. In multi-agent information systems, a preference is (meta) knowledge about what types of information have utility for a requester, both in form (John follows the price of SUNW) and other characteristics (John wants only free information; John wants stock quotes at least every 35 minutes). A capability is (meta) knowledge about what types of requests can be serviced by a provider (Mary can provide the current price of any NASDAQ stock, 15 minute-delayed, for free at a rate of 10 quotes per minute). While this paper will focus on information providing agents, providers might also accomplish other types of tasks.

From a privacy standpoint, preference information can flow from a requester to a provider, and capability information can flow the other way. Agents that deal with preference or capability information that are *neither* requesters *nor* providers (from the standpoint of the transaction under consideration) we call *middle-agents*. Different organizational solutions to the connection problem stop this flow of information at different points. We will discuss the full scope of design possibilities presented by our model and a full experimental implementation of two possible designs. In one, known variously as *matchmaker*, *yellow pages*, or *directory agent* systems [Genesereth and Ketchpel, 1994; Finin *et al.*, 1994; Kuokka and Harada, 1995], capabilities come to be known by all including the requesters, but preferences are kept initially private. In the other, which we call in general *brokered* systems, only the broker comes to know both preferences and capabilities of a class of requesters and providers. Market-based programming systems [Wellman, 1993] are a popular subclass of brokered system.

Privacy, however, is only one concern when choosing a solution to the connection problem. A designer also needs to consider other characteristics, such as the efficiency with which requests are handled and resources are used, the vulnerability of the system to the failure of some component, and the ability to quickly adapt as an open system to changing preferences and capabilities. Our ongoing research aims to develop empirically-validated models of the relationships between the various performance characteristics and system parameters.

This paper will first present our privacy based model of the connection problem, and briefly lay out the space of organizational solutions and their characteristics. We then briefly describe how matchmaking and brokering behaviors can be defined in terms of well-understood communicative acts. By defining these behaviors and how they interact, we are able to better understand how our space of solutions to the connection

*This work was supported by ONR Grant #N-00014-96-1-1222.

preferences initially known by	*Capabilities initially known by*		
	provider only	provider + middle agent	provider + middle + requester
requester only	(broadcaster)	"front-agent"	matchmaker/yellow-pages
requester + middle agent	anonymizer	broker	recommender
requester + middle + provider	blackboard	introducer/bodyguard	arbitrator

Table 1: Middle-agent roles in the solution space to the connection problem, categorized by explicit initial privacy concerns.

problem constrain agent architecture and behavior design.

Finally, we will examine some empirical results on the comparative performance of brokered and matchmade systems. The questions we will be examining include first the quantitative end-to-end response time advantages and disadvantages of matchmaking and brokering behavior. Second, we will examine characteristics of these behaviors with respect to robust and adaptive open systems, where agents might enter and exit the system at any time. Our experimental results were achieved using an implementation of the WARREN multi-agent financial portfolio management system [Sycara *et al.*, 1996; Decker *et al.*, 1996].

2 Definitions and Model

We will examine the connection problem from the standpoint of privacy considerations. In particular, we examine knowledge about requester agent preferences, and provider agent capabilities. A specific *request* is an instance of an agent's preferences, and a specific *reply* or action in service of a request is an instance of an agent's capabilities. Furthermore, an agent can have a mental state with respect to a particular specification of a preference or capability. An *advertisement* is a capability specification such that the agent creating the advertisement is committed to servicing any request that satisfies the advertisement's constraints. Symmetrically, a *want-ad* is a preference specification by a requester who is committed to accepting any reply that meets the constraints in the preference spec.

Preference information can initially be kept private at the requester, be revealed to some middle agent (neither the requester nor the ultimate provider), or be known by the provider itself. The same three possibilities exist for capability info (see Table 1). This leads to nine general middle-agent roles in information-gathering organizations, of which this paper has room to examine three. A **blackboard** is a middle-agent that keeps track of requests. Requesters post their problems; providers can then query the blackboard agent for events they are capable of handling. This class includes newsgroups and bulletin boards. A **broker** is a middle-agent that protects the privacy of *both* the requester and provider. The broker understands both the preferences and capabilities, and routes both requests and replies appropriately. Neither the requester nor provider ever knows directly about the other in a transaction. A **matchmaker/yellow-pages** is a middle agent that stores capability advertisements that can then be queried by requesters. The requesters then choose and contact any provider they wish directly.

Clearly whenever a requester and provider come into direct contact multiple times it is possible for one to learn the preferences or capabilities of the other. For example, some work describe agents that attempt to learn the preferences of people [Maes and Kozierok, 1993]. Other work reports on agents that learn capabilities [Kautz *et al.*, 1996]. Hybrid organizations can be used to protect both requesters and providers from learning agents; e.g., requesters can use an anonymizer middle-agent even if the nominal organization is blackboard-like. An anomynizer middle-agent knows the preferences of the requester, posting on the requester's behalf (such systems are really used on the Internet). Such hybrids add to communication costs and potential for failures, however. We will later discuss another hybrid, the combination of matchmade and brokered organizations, that avoids some of these problems and brings together robustness and efficiency characteristics.

We will focus our attention on the 3 diagonal boxes representing minimal information travel: the blackboard organization where preference commitments or individual requests are posted for everyone to see, but true capabilities remain hidden; a matchmaker organization where providers freely advertise their capabilities but requesters remain private; and brokered organizations where preferences/requests are joined to capabilities using either market allocation mechanisms, OS load balancing algorithms, or managerial task assignment schemes. Because our interest is in open systems, where all agents may come and go, and preferences and capabilities may change dynamically, we are limiting our discussion to middle-agent systems where the capabilities of all agents are not known and hard-coded at compile time.

2.1 Organizational Role Behaviors

Each organizational model relies on three basic roles: that of the requester, the middle-agent, and the provider. Any one agent in a domain system might take on multiple roles, for example an agent that requests basic info from several providers, does some complex integration, and then serves the integrated info to other requesters. In this model, communicative acts are limited to requests, replies, and commitments. This has two benefits: first, the semantics of requests and commitments are well-understood [Finin *et al.*, 1994; Cohen and Levesque, 1995], and second, such a model allows us to build simpler agents that can work in an open environment with hybrid behaviors (e.g., both matchmaking and brokering).

In a matchmade organization, providers *advertise*[1] their capabilities with a matchmaker[2]. If those capabilities change, or the agent exits the open system, the provider *unadvertises.* A matchmaker stores these advertisements in a local database. A requester wishing to ask a query first formulates a meta-query asking for advertisements from agents that could respond to the query. This meta-query is asked of a matchmaker, which responds with a set of matching advertisements. The requester can then use its full preferences to choose a provider, and make its request directly. Furthermore, if this type of query is asked often, then the requester can *subscribe* to updated advertisements from a matchmaker, and keep a local cache of the current advertisements (see Section 2.3). Blackboard organizations are extremely similar, but with requester and provider behavior reversed (e.g. requesters send *want-ads* to the blackboard). In brokered organizations, requester behaviors remain the same. In a *pure* brokered organization the brokers are generally known by all the agents, just like a matchmaker is. However, for practicality in an open system *hybrid* brokered organizations use a matchmaker so that providers and requesters can find the appropriate broker (see Section 2.5). Providers query a matchmaker to find an appropriate broker, and then advertise with one broker. Brokers advertise summary capabilities built from all the providers that have advertised with them; these capabilities are advertised in turn to the matchmaker. When a request comes in, the broker matches it with a provider and sends it on; the reply is then sent back to the original requester. The methods by which a broker assigns requests to providers can drawn from several areas of research, including OS load balancing, managerial task assignment, and market-based economics. Which methods to choose will depend on several environmental factors (such as whether the providers are self-interested or cooperative agents) and will be a subject for future work.

2.2 Efficiency

The main performance attribute which we have measured (Section 3) is r, the total elapsed time taken by a requester to satisfy a service objective. It includes: (1) time spent planning and scheduling by the requester, middle-agent, and provider, S (2)time spent communicating between agents (given that we always use a middle-agent rather than compiling in fixed agent name, this feature of our agents, denoted C, includes four communication actions); (3) time spent by the provider providing the service T; (4) time spent waiting at a provider which is busy fulfilling prior requests, denoted Q (this is a function of the request generation period, P, and of the number of providers, N).

Our system can be roughly described by a queuing network model [Lazowska *et al.*, 1984]. According to queuing network theory, the total elapsed time to fulfill a request is $r = D + Q$, where D is total computational demand of the request (in our case, $D = T + S + C$). Note that r, like Q, is a function of the request generation period and of the number of providers. If requests are generated at a rate greater than the maximum system throughput, i.e. if $P < \frac{D}{N}$ then the system will be *saturated* and r will grow without bound. Otherwise, a fundamental result of queuing theory is that the expected elapsed time per request is:

$$r = \frac{D}{1 - \frac{D}{PN}} \tag{1}$$

This result depends on the service request load being equally distributed across all providers, or else the elapsed time per request will be greater. Since a brokered system can precisely balance the load on providers, while a matchmaker or blackboard organizations only stochastically does so, we would expect the broker to provide better elapsed times. Of course some decentralized load balancing can be done even in a matchmade organization, at the expense of extra communication.

2.3 Robustness

Decentralized organizations such as matchmade or blackboard orgs with caching are significantly more robust than centralized organizations such as brokered orgs. Malone [Malone, 1987] examines the basic vulnerability of decentralized and centralized markets, defined as the sum of the expected costs of each possible failure times the probability of that failure. Let p_r, p_m, and p_p be the probability of failure of a requester, middle-agent, and provider respectively; and C_p be the cost of reassigning a task upon provider failure, C_r be the cost of losing all the requests at a single requester (this is a function of r and P above), and C_m be the cost of losing all access to a single class of capabilities. Assuming N providers, A requesters, and B brokers each brokering a single class of capabilities, the vulnerability of a brokered organization is

$$Ap_rC_r + Bp_mC_m + Np_pC_p$$

Interestingly, a matchmade system where every requester must check the matchmaker/yellow-pages every time, does no better! However, it is easy to have a requester cache matchmaker results so that if a matchmaker fails, the cached information can be used temporarily. This behavior changes the vulnerability of a matchmade system to

$$Ap_rC_r + Np_pC_p$$

which agrees with Malone's predictions for decentralized markets. Blackboard systems are similar to the matchmade systems, but it is more unlikely for cached *requests* to be useful, as opposed to cached *capabilities.*

2.4 Adaptivity

The final characteristic we will discuss is the effect of dynamically changing preferences and capabilities. An example of a preference change in the WARREN domain is the need to track a new stock, or an old stock at a new frequency. An example of

[1] All communications here are done via the appropriate KQML performatives [Finin *et al.*, 1994].

[2] At some fixed location; there could be more than one; this can be handled like the Internet DNS.

a capability change is the entry or exit of a new provider. The ability of an organization to quickly adapt to new preferences or capabilities is a function of the distance that the information has to travel, and the costs of keeping that information up-to-date. For example, in a matchmade organization, capability information is available locally to the requesters, and so a change in preferences can be acted on instantaneously. The reverse holds for blackboard organizations and changing capabilities. In either case, the primary costs are in keeping the local cache of public information (capability info, in the matchmade case). Brokered organizations again represent a useful design alternative, requiring only that the broker be notified of changes, but requiring such messages for both capability and preference changes.

Since we are interested in an open system, let us look at what happens to the maximum service time R when providers come and go. R will obviously change as the number of providers N changes, but Eqn 1 only holds when the system is not saturated. When the system *is* saturated, the queues of the remaining agents will begin to grow (along with R) until the lost providers are restored. If we define the excess capacity of the system at time t as:

$$\mathrm{Ex}(t) = \frac{N(t)}{D(t)} - \frac{1}{P(t)} \tag{2}$$

then the maximal queue length MQL at time t is:

$$\mathrm{MQL}(t) = \max(0, \mathrm{MQL}(t-1) - \mathrm{Ex}(t)) \tag{3}$$

and we can predict the maximal response time as the max of the steady state response time and the response time implied by the maximal (FIFO) queue length. As usual, this only holds for a brokered system where the load is perfectly balanced as providers come back on line. In a matchmade system, if the agents make no attempt at load balancing, the MQL may continue to grow even after providers come back on line, and when it falls it will do so more slowly than in the brokered system (Section 3).

2.5 Hybrid Organizations

Real solutions to real world problems will often require hybrid approaches. If capabilities and matching preferences can be clearly partitioned (creating different connection problem classes) then each class can use a different organization simultaneously. For example, how do agents find an appropriate broker? If we ignore precompiled solutions, one way is to use a matchmaker. Providers advertise their capabilities privately to a broker, and the broker in turn advertises a (usually) more abstract capability(ies) to a matchmaker.

Another way to use hybrid MM/Broker organizations is to take advantage of the best characteristics of each organization alone, at the cost of reduced privacy in emergency situations. Brokered organizations have higher efficiency because of their centralized load-balancing, but are critically non-robust to broker failure. By switching automatically to a matchmade organization (and thus losing provider privacy) the system as a whole becomes robust to broker failure.

Due to our careful definitions of roles earlier, such a hybrid system requires only a small change to the behavior of provider and broker agents: brokers must advertise their brokering as a capability to a matchmaker, and providers must upon initialization register with the matchmaker for broker-capability updates. When a broker is available, the provider uses it; if not, the provider advertises with a matchmaker until one becomes available. A broker failure (once detected via a KQML "Sorry" or timeout) can then transform into a matchmade organization.

3 Performance Tradeoffs

The decision to use matchmaking or brokering to solve the connection problem offers many performance tradeoffs. In our remaining space we present two short representative experiments.

System performance is dependent upon a large number of parameters, including the rate at which service requests are generated; the number of providers in the system; the time needed by each provider to fulfill a request; agent failure rates, and so on. We will consider two alternative systems. Each consists of some number of homogeneous providers and requesters. All agents run on serial processors, and the basic service action is non-interruptible. In the "Matchmade" system, each provider advertises itself to one *matchmaker* agent. Requesters query the matchmaker to obtain a current list of providers, choose one randomly, and send it a service request. In the "Brokered" system, providers advertise themselves to a distinguished *broker* agent. Requesters send all service requests directly to the broker, who farms them out to the providers—seeking to equalize the load among them.

Our implementation of these systems consists of real, implemented agents, who experience real communication and processor latencies, etc. These agents are part of the WARREN multi-agent financial portfolio management system. [Sycara *et al.*, 1996; Decker *et al.*, 1996] The broker and matchmaker are the same agents used in the actual portfolio management system. Providers and requesters are instances of WARREN providers and requesters, but we have standardized on a single abstract service to be provided (modeled after stock ticker services).

The actual service time for each request and the period between requests are generated randomly; the service time is distributed normally around the mean T, and the request generation period is distributed exponentially around the mean P. But because we are using real rather than simulated agents, some parameters are beyond our control, such as the inter-agent communication latency, the computational needs of the broker or matchmaker, and the amount of time spent by the providers and requesters on planning, scheduling, and other internal operations.

We make some further assumptions about the ranges of values that our system parameters will take on. First, we assume that the service time is relatively long compared to the computational overhead of the matchmaker, broker, and providers themselves. This is consistent with actual WARREN agents, which typically require 30 seconds or more to access Internet

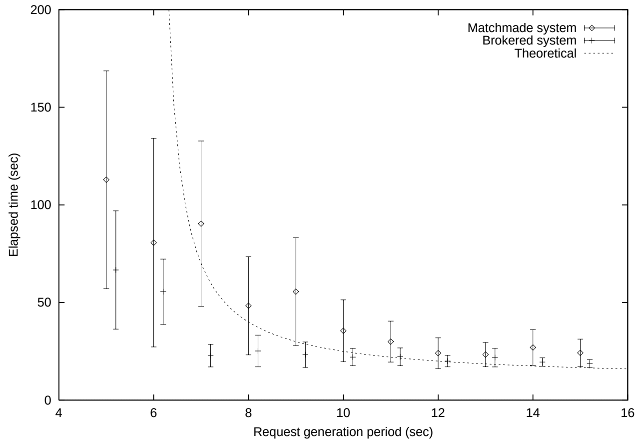


Figure 1: Mean time for 100 service requests as a function of request generation period

resources. Second, we assume that the number of providers is relatively small (we have experimented with systems of up to 20 providers), which is again consistent with the operational WARREN system.

3.1 Experiment One: Response Time

In our first experiment, we validate the theoretical model of Eq. 1, and empirically compare the brokered and matchmade systems. For a fixed service time T and number of providers N, we vary the request generation period P. For each period, we generate 100 requests, and measure the mean and standard deviation of their elapsed times. Figure 1 shows the results as the request generation period varies between 5 and 15 seconds, for systems with 3 providers and a service time of 15 seconds.

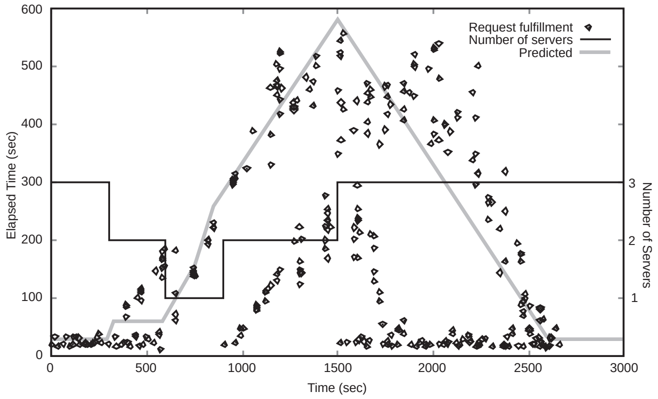
Note that despite the crudeness of Eq. 1, it gives a good indication of the expected response time, especially for larger request generation periods[3]. It is clear that the load balancing of the brokered system confers a response time advantage over the matchmade system.

3.2 Experiment Two: Provider Failure and Recovery

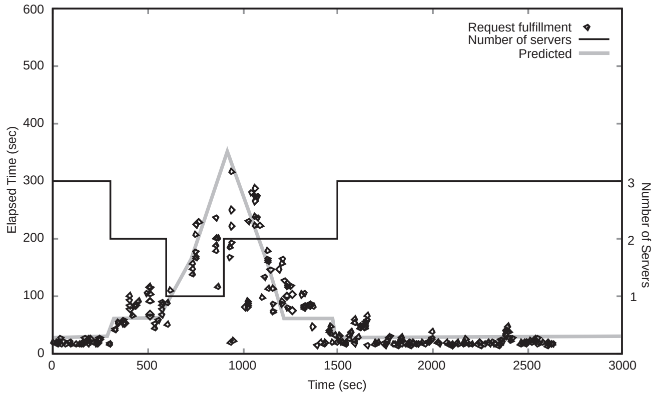
In our second experiment, we investigate the effect of provider failure and recovery on our two systems. We begin with three providers, and fix the service time and request generation period at 15 and 10 seconds, respectively. After five minutes, we kill one of the providers, and after five more minutes, we kill a second one. Five minutes after that, we bring one of the servers back on line, and then ten minutes later the third one returns. When a provider dies, it sends a SORRY message for each outstanding request. Each of these requests must be reallocated (by either the broker or the original requester) to another provider.

[3]For shorter periods, when the system is more highly loaded—or even saturated—our measured values fall below the predictions because we are performing only 100 queries. The earlier requests experience less queuing time, and so skew the results downwards.

Figure 2 shows the results of this experiment. Each point represents the completion of a service request. The response-time superiority of the brokered system stems from the difference in behavior of the two systems when the failed providers come back online. When there is only one provider left operating, the system is saturated, so that provider begins to build up a large backlog of requests. When the second and third providers become available again, the requesters in the matchmade system continue to allocate one-half or one-third of their requests to the overloaded provider, so the backlog persists for a long time.[4] In the brokered system, on the other hand, all new requests are allocated to the new provider, allowing the backlog at the congested provider to quickly dissipate.



(a) In a matchmade system



(b) In a brokered system

Figure 2: Predicted and actual effect of provider failure and recovery

Figure 2 also shows the predicted response time (for the maximally loaded provider) in the brokered organization as derived from Eqs. 1, 2, and 3 in Section 2.4. Regression analysis shows that the predicted response matches the actual response of the maximally loaded provider with $R^2 = 0.77$ (77% of the variance explained).

Of course this prediction does not hold for the matchmade

[4]This effect could be reduced if requesters make an effort at active load balancing.

system. Rewriting Eq. 2 for matchmade organizations as a recurrence equation $\mathrm{Ex}_M(t) =$:

$$\begin{cases} \mathrm{Ex}_M(t-1) & N(t) = N(t-1) \\ \frac{N(t)}{D(t)} - \frac{1}{P(t)} & N(t) < N(t-1) \\ \mathrm{Ex}_M(t-1) + \frac{1}{N}(\frac{N(t)}{D(t)} - \frac{1}{P(t)}) & N(t) > N(t-1) \end{cases}$$

This is the same as before except when adding new agents after being saturated, in which case the maximal queue length is reduced by an amount proportional to the optimum experienced by the brokered system. Using this excess capacity equation results in an regression R^2 value of 83% with respect to the measured experimental data.

4 Conclusions

This paper examined solutions to the agent connection problem using third-party "middle-agents." Using privacy as an organizing paradigm, we outlined 9 alternative types of middle-agents that result in corresponding organizational classes. We examined analytically the efficiency, robustness, and adaptiveness characteristics of three common alternatives. Finally, we validated two of our models using experimental implementations for matchmade and brokered organizations in a real multi-agent system, concentrating on the speed at which the alternatives adapt to provider failures and reappearances.

Although we did not have space to discuss all 9 alternatives, we hope to stimulate discussion about, and work on, these alternate approaches, each applicable to different situations. Although the concepts of matchmaking, recommending, and brokering have been described in general elsewhere, only Kuokka has provided any detailed experiences with matchmaking. This paper outlines a much larger playing-field for middle-agents, and includes implementation experiences with both matchmade and brokered systems. This is the first work to analyze some of the detailed characteristics of these organizations, drawing on queuing theory and Malone's modeling work with human organizations.

Matchmaker organizations can become elegant decentralized markets with the proper caching mechanisms. They offer preference privacy to requesters, and each requester keeps total control over its own control decisions (so adapting to changing preferences is immediate). Each agent needs to be smart enough to construct a meta-query and evaluate the resulting alternative provider choices. With local caching, matchmade organizations are highly resistant to total failure, but rather degrade gracefully. No simple optimal load balancing is possible; we described an analytical model that predicts system response times without load balancing. Using a matchmaker has slightly higher overhead due to the extra queries.

Brokered organizations include centralized economic markets or traditional bureaucratic managerial units. They can also handle the dynamic entry and exit of agents, and provide an easy way to do load balancing. They can protect the privacy of both requester preferences and provider capabilities. However, they suffer from the need of agents to have static knowledge of the brokers (in a pure, non-hybrid system). In addition, a broker is a communication bottleneck (since all requests and replies need to go through the brokers). Worst of all, they form a single point of failure that cannot be mitigated via local caching.

The solution that we are currently working on is a hybrid system with both matchmakers and brokers. By design, our specified agent behaviors work interchangeably with both organizational roles. A hybrid system allows us to capitalize on the lower overhead and efficient load balancing of a brokered system while retaining the dynamic naming capabilities and greater robustness of a matchmaker system. By trading off privacy in an emergency, a hybrid system can continue to function even in the face of broker failure.

References

[Cohen and Levesque, 1995] P.R. Cohen and H.J. Levesque. Communicative actions for artificial agents. In *Proc. ICMAS-95*, pages 65–72. AAAI Press, June 1995.

[Davis and Smith, 1983] R. Davis and R. G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, January 1983.

[Decker *et al.*, 1996] K. Decker, A. Pannu, K. Sycara, and M. Williamson. Designing behaviors for information agents. In *Proc. Autonomous Agents 97*, pages 404–413. ACM Press, February 1997.

[Finin *et al.*, 1994] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an agent communication language. In *Proc. CIKM-94*. ACM Press, November 1994.

[Genesereth and Ketchpel, 1994] M.R. Genesereth and S.P. Ketchpel. Software agents. *CACM*, 37(7):48–53,147, 1994.

[Kautz *et al.*, 1996] Henry Kautz, Bart Selman, and Al Milewski. Agent amplified communication. In *Proc. AAAI-96*, pages 3–9, August 1996.

[Kuokka and Harada, 1995] D. Kuokka and L. Harada. On using KQML for matchmaking. In *Proc. ICMAS-95*, pages 239–245. AAAI Press, June 1995.

[Lazowska *et al.*, 1984] E.D. Lazowska, J. Zahorjan, G.S. Graham, and K.C. Sevcik. *Quantitative System Performance.* Prentice Hall, 1984.

[Maes and Kozierok, 1993] P. Maes and R. Kozierok. Learning interface agents. *Proc. AAAI-93*, pages 459–465, July 1993.

[Malone, 1987] Thomas W. Malone. Modeling coordination in organizations and markets. *Management Science*, 33:1317–1332, 1987.

[Sycara *et al.*, 1996] K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng. Distributed intelligent agents. *IEEE Expert*, 11(6):36–46, December 1996.

[Wellman, 1993] Michael Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *JAIR*, 1:1–23, 1993.