# Nomad: *Mobile Agent System for an Internet-Based Auction House*

TUOMAS SANDHOLM AND QIANBO HUAI
*Washington University*

Nomad is the mobile agent system integrated with eAuctionHouse, our next-generation Internet auction server. With the Nomad system, mobile agents travel to the eAuctionHouse site and participate in auctions on the user's behalf. Users can create agents using Java or can automatically generate agents from Nomad's template agent library.

As the Internet moves into the mainstream, electronic commerce is becoming an important mechanism for conducting business. It helps merchants and consumers reduce business costs and enables customized delivery of goods and services. Among the current business models, electronic auctions are emerging as one of the most successful e-commerce technologies.

There are several successful commercial Internet auction sites, such as eBay and Yahoo, as well as interesting academic Internet auction houses.[1] Our motivation in developing an auction server, eAuctionHouse, was to prototype next-generation features and test their feasibility, both computationally and in terms of consumer ease of use. eAuctionHouse is to our knowledge the first, and currently only, Internet auction site that supports combinatorial auctions,[2–4] bidding via quantity-price graphs,[5] and mobile agents. eAuctionHouse acts as a third-party auction site, allowing users across the Internet to buy and sell goods and to set up markets. eAuctionHouse is available for testing at http://ecommerce.cs.wustl.edu.

As in conventional Internet auctions, in eAuctionHouse a user visits the auction website to create or close an auction or to submit bids. However, eAuctionHouse supports two additional mechanisms for creating auctions, closing auctions, and bidding: a user can send a formatted text string directly through a TCP/IP connection, or use Nomad, the integrated mobile agent system.

Another article presents a detailed view of eAuctionHouse.[5] This article focuses on the Nomad system.

## MOBILE AGENT SYSTEM FOR ELECTRONIC AUCTIONS

Nomad allows mobile agents to travel to the eAuctionHouse site and actively participate in auctions on the user's behalf even when the user is disconnected from the network. This reduces network traffic and latency, and the agents can respond to changes in the auction quicker than remote users could. The speed of executing a computationally intensive bidding strategy may also increase when agents execute on a powerful server. Mobile agents need not necessarily be bidding agents. They can be used for collecting information, learning price distributions, or setting up auc-

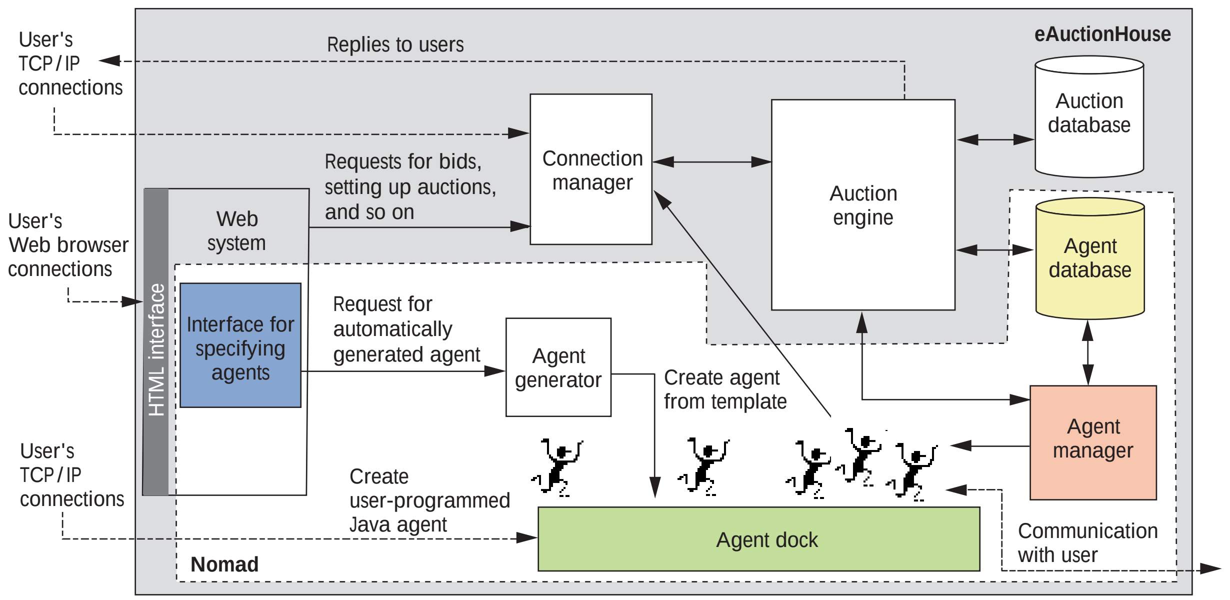

eAuctionHouse
User's TCP/IP connections
Replies to users
Requests for bids, setting up auctions, and so on
Connection manager
Auction engine
Auction database
Agent database
User's Web browser connections
Web system
HTML interface
Interface for specifying agents
Request for automatically generated agent
Agent generator
Create agent from template
Agent manager
User's TCP/IP connections
Create user-programmed Java agent
Agent dock
Communication with user
Nomad

**Figure 1. Nomad architecture within eAuctionHouse. The main components of the Nomad system are the interface for specifying agents, the agent dock, the agent manager, and the agent database.**

tions. (For the advantages of using mobile agents, see the sidebar, "Mobile Agents in Internet-Based Auctions").

When multiple distributed eAuctionHouses are installed across the network, multiple Nomads help to form a virtual electronic auction site network. To compare deals across different eAuctionHouses, however, an agent does not necessarily have to migrate—from one agent dock it can download and upload information from all of the eAuctionHouses through their TCP/IP connections. The agent can make the decision to migrate dynamically based on the amount of information transmitted, latency, and so on.

Also implemented in Nomad is a mobile agent control scheme. After registering itself at the server, a mobile agent can be seen and managed in its creator's user portfolio. Once an agent docks on a server, it registers itself on the server. When the user asks, the server displays all of the user's registered agents. If, for example, the user asks to kill an agent, the server sends a message to the agent. The agent then unregisters itself and the system deletes the registry entry. If a user were to program an agent that leaves the server without unregistering, the registry entry would remain. If the user then asked to kill the agent, the system would delete the registry entry without actually killing the agent because the system does not keep track of where agents have migrated. In summary, in the current implementation we do not provide automatic agent tracking beyond a single agent dock, but leave that to the programmer of the agent.

The high-level architecture of a Nomad is illustrated in Figure 1. A Nomad system consists of four main components:

- an interface for specifying agents
- an agent dock
- an agent manager
- an agent database

The eAuctionHouse Web system provides the HTML interface for users to navigate within eAuctionHouse. When a user sends a request for automatically creating a mobile agent via the Web system, the request is directed to an agent generator. All other requests are forwarded to the connection manager. Figure 1 shows the connection manager receiving input from three kinds of sources: the Web system, TCP/IP connections, and agents.

Internally the connection manager does not distinguish between these sources. The same TCP port is used for communication and all requests are sent as formatted text strings. When a request is accepted, the connection manager initiates a handler thread. The handler checks the request's validity, takes necessary actions, synchronizes accesses to the database, retrieves and updates the database, and returns a result. Conceptually, each request is mapped to a particular handler algorithm. The modules and functionality of the auction engine itself are presented elsewhere.[2,5]

Part of the Web system is dedicated to specifying agents. Any request through that part is sent to the agent generator, which then decodes the formatted text string, creates mobile agents following those instructions, and launches them onto the agent dock. Mobile agents reside and execute on the agent dock.

We use the Concordia system (http://www.meitca.com/HSL/Projects/Concordia) as the basis of our agent dock. Concordia is a framework for developing and managing mobile agent applications.[6] It is a Java application and supports mobile agents written in Java. Application interfaces are provided in Concordia for sending agents around the network. In Nomad these are used both for launching automatically programmed agents and for sending agents manually programmed by users.

Concordia agents process data at the data source. Network transport is hidden from applications, developers, and users. Typically, a Concordia agent has an itinerary, which can be seen as a list of network destination addresses. Associated with each address is an action—a Java class method executed when the agent travels to the associated site. The

## Mobile Agents in Internet-Based Auctions

The use of agents in electronic auctions has several advantages for the user:

- An agent can monitor the auction events the user has deemed relevant. When such events occur, the agent can alert the user. This frees the user from having to poll the auction repeatedly.
- Compared with traditional bidding where every bid is specified parametrically (for example, by the amount the user bids), bidding agents give the user more flexibility when customizing a bidding strategy. The strategy can be a function of time, of other participants' bids, and so on.
- The agent can also be programmed to monitor external events, and to condition its bidding on those events (for example, stock prices or news). In other words, the agent can make decisions based on all available information that the bidder considers relevant.
- Prototypical bidding agents can be analyzed game-theoretically off-line so that they will bid optimally on the user's behalf in given auction settings. This puts expert bidders and amateurs on a more equal footing for ecommerce: since the bidder agent will optimally bid on the user's behalf, the user need not engage in strategic considerations when revealing preferences to the agent.
- Agents can be built to track bids in multiple auction houses, looking for the best deal and/or coordinating the user's bids in the different auctions. For example, an agent can submit bids to multiple auction sites for the same item, but at any time allow at most one of the bids to be winning (highest within that auction). While this is possible for a human user to do without an agent, an agent saves effort, and makes such behavior viable even in settings where the user's time is costly.

There are additional advantages that stem from agents' execution on (or near) the auction server.

- In many current Internet auctions, most of the bidding activity occurs just before the auction closes. With agents that execute on the server side, the user can avoid the network lag of getting the most current information from the auction to the user's site, and of bids traveling from the user's site to the auction.
- Agents that execute on the server side will continue to operate as usual even when network connections are down or slow.
- The user does not need to be continually connected to the network. For example, a user can connect a laptop to the network via a phone link on an airplane, launch an agent to execute on the auction server or on a home computer, and disconnect.
- If the information transferred between the agent and the auction exceeds the code size of the agent, sending an agent to execute server-side uses less bandwidth than client-side execution. When the agent communicates locally at its destination rather than over the network, network traffic and latency are reduced because the amount of data transferred around the network is reduced.
- In simple auctions with infrequent activity, the information downloaded from the auction to the agent (other bids, quotes) and the information that is uploaded from the agent to the server (mainly bids) might not exceed the size of the agent. However, in highly active auctions and in combinatorial auctions, the information that is transferred can easily exceed the size of the agent. In a combinatorial auction, bids can be submitted on combinations of items. So, if quotes are provided, they need to be provided on combinations of

itinerary can be altered dynamically during the agent's trip. Agents can also collaborate with the help of an event-distribution mechanism and other services.

The agent manager notifies agents when the auction information they are interested in is altered. The agent database stores information about agents—such as their creators and the information they want to receive. By communicating with the agent manager, agents can not only utilize the event-distribution mechanism (which triggers an agent only if something of interest happens in the auctions, rather than requiring the agent to poll the auctions), but they can also be seen via the eAuctionHouse Web system. This gives users a convenient interface for managing their agents. With this mechanism, the agent programmer does not have to write code for the agent to handle communication with the user and act based on that communication, although the user can do this to implement additional functionality.

## GENERATING MOBILE AGENTS

Nomad supports the creation of mobile agents by allowing users to program their own agents or launch parameterizable template agents that have been designed and programmed in advance.

items and the number of combinations is exponential (of course, one could provide quotes on select combinations only). Furthermore, a new bid on a combination generally changes the quote on a large number of combinations, further increasing the amount of quote information that an agent needs to download.

The benefits of remote execution could be captured by either remotely executing nonmobile agents or by mobile agents. Other advantages are specific to mobile agents.

- Mobile agents can potentially take advantage of the available services distributed across the network. For example, they could travel to and execute on powerful servers with excess CPU time and disk space. This can be pertinent for bidder agents if, for example, their bidding strategies include complex computations such as statistical analysis and projection.
- The use of mobile agents can lead to more effective load balancing. A mobile agent can move to an agent dock where the load is currently not too high. This leads to faster execution of the agent's bidding strategy.
- Mobile agents can react to network latencies that vary over time in different parts of the network. An agent can move to an agent dock with a less congested connection to the auction server.
- Using mobile agents, the decision of local versus remote execution can be made dynamically at run time based on the volume of information being uploaded and downloaded, network latency and congestion, and the availability, speed, and cost of computation at different sites.

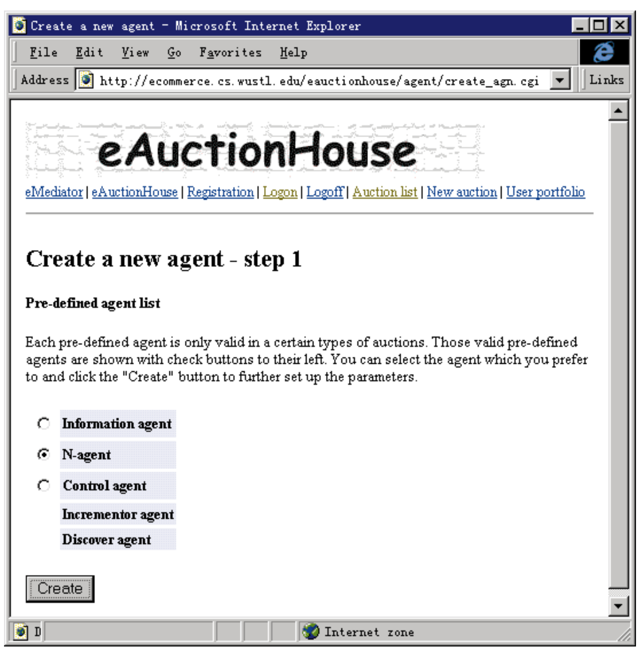
Naturally, there are also disadvantages to using mobile agents. Most notably, the appropriate allocation of resources—such as CPUs, RAM, and disk—among the mobile agents of different users remains an open research area.[1]

Several auction sites have solutions other than mobile agents. eBay has a proxy bidder "agent" that allows the user to enter a reservation price. As long as the auction is open and the user's reservation price has not been reached, the agent bids the minimum amount necessary to become the highest bidder. However, such an agent limits the user's choice of bidding strategy. For example, when a user's valuation of an item depends on other bids, such a simple agent is no longer optimal; rather, the agent should update its valuation dynamically based on the other bids so far. This involves taking into account the effect of the winner's curse: if the bidder bids the perceived valuation of the item and wins, the bidder will know that he or she paid too much because others valued the item less.[2] Furthermore, the simple proxy bidder agents offered by current Internet auctions do not allow the user to coordinate bids across multiple auction houses automatically.

The Michigan Internet AuctionBot could be viewed as supporting agents in the sense that it provides a TCP/IP-level message protocol by which agents can participate in the auction.[3] However, no support is provided for mobile agents.

### References

1. J. Bredin, D. Kotz, and D. Rus, "Market-based Resource Allocation for Mobile Agents," Proc. Second Int'l Conf. Autonomous Agents (AGENTS), ACM Press, New York, May 1998.
2. P. Milgrom, "Auctions and Bidding: A primer," *J. Economic Perspectives*, Vol. 3, No. 3, 1989, pp. 3–22.
3. P.R. Wurman, M.P. Wellman, and W.E. Walsh, "The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents," *Proc. Second Int'l Conf. Autonomous Agents (AGENTS)*, ACM Press, New York, May 1998, pp. 301–308.


Create a new agent - Microsoft Internet Explorer

Address http://ecommerce.cs.wustl.edu/eauctionhouse/agent/create_agn.cgi

eAuctionHouse

eMediator | eAuctionHouse | Registration | Logon | Logoff | Auction list | New auction | User portfolio

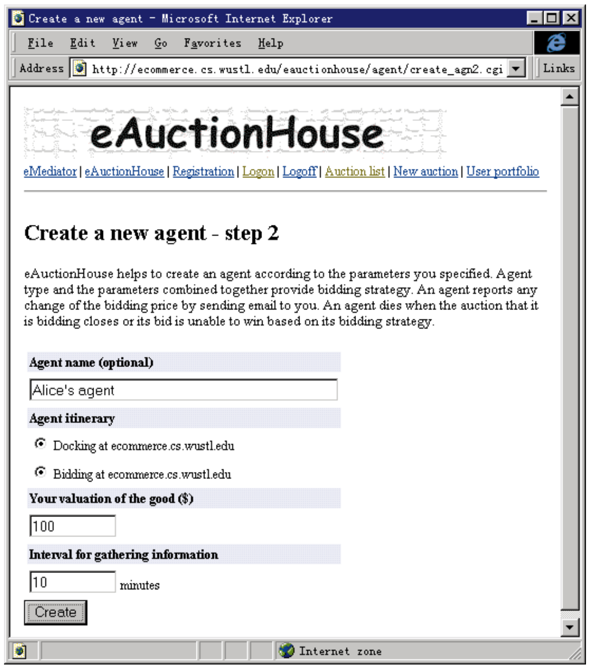
**Create a new agent - step 1**

**Pre-defined agent list**

Each pre-defined agent is only valid in a certain types of auctions. Those valid pre-defined agents are shown with check buttons to their left. You can select the agent which you prefer to and click the "Create" button to further set up the parameters.

- ○ Information agent
- [✓ N-agent]
- ○ Control agent
- Incrementor agent
- Discover agent

Create


Create a new agent - Microsoft Internet Explorer

Address http://ecommerce.cs.wustl.edu/eauctionhouse/agent/create_agn2.cgi

eAuctionHouse

eMediator | eAuctionHouse | Registration | Logon | Logoff | Auction list | New auction | User portfolio

**Create a new agent - step 2**

eAuctionHouse helps to create an agent according to the parameters you specified. Agent type and the parameters combined together provide bidding strategy. An agent reports any change of the bidding price by sending email to you. An agent dies when the auction that it is bidding closes or its bid is unable to win based on its bidding strategy.

**Agent name (optional)**
Alice's agent

**Agent itinerary**
[✓ Docking at ecommerce.cs.wustl.edu]
[✓ Bidding at ecommerce.cs.wustl.edu]

**Your valuation of the good ($)**
100

**Interval for gathering information**
10 minutes

Create

**Figure 2. Creating a mobile agent in eAuctionHouse. Step 1 of agent creation is to choose an agent type from a set of prototype agents that the system has determined to be appropriate for the auction type in question. Step 2 of agent creation is to set parameters for the agent.**

## Tailored Agents

Users can program their own mobile agents in Java, which allows maximal flexibility in what agents can do. The agents can use highly tailored bidding strategies that consider input from what is transpiring in the auction, other auctions, and in the news. The agents can also use highly tailored migration schemes. The user programs the agent to communicate with eAuctionHouses through TCP/IP connections using a string format that we have specified to allow rich forms of bidding, collecting information, and setting up auctions and markets.

## Template Agents

To speed agent generation and to enable nonprogrammers to create mobile agents, the Nomad system allows automated generation of agents based on HTML forms. Using the forms, the user chooses from a library of preprogrammed template agents that are recommended to the user based on the auction type in question. Nomad currently makes the following parameterizable mobile agents available for automated generation:

- The *information agent* monitors an auction and sends e-mail to the user when specified events occur. With this agent, the user does not have to poll the auction and is notified of important events immediately.
- The *incrementor* agent implements the dominant strategy on the user's behalf in single-item, single-unit, ascending open-cry first-price private value auctions (that is, English auctions). It bids a small amount more than the current highest bid, and stops if the user's reservation price is reached. With this agent, the user does not have to follow the auction. The user's dominant strategy in these settings is to report the valuation truthfully to the agent. Not accounting for the technical aspects, such as having its own thread of execution and being able to migrate, this agent provides the same functionality as current proxy bidder "agents" like those on eBay.
- The *N-agent* is for single-item, single-unit, sealed-bid first-price auctions where the number of bidders, *N*, is known, and the bidders' private valuations are independently drawn from a uniform distribution. The goal in this type of auction is to win the auction but to underbid strategically so as to minimize payment. This involves guessing what others will

bid. The symmetric Nash equilibrium strategy is to bid the user's valuation times $(N-1)/N$.[7] Since this is how the $N$-agent bids, the user is motivated to reveal the true valuation to the agent.

- The *control agent* submits very low noncompetitive bids. This agent is a speculator's tool that artificially increases the number of bidders so as to mislead other bidders, such as the $N$-agent. For example, a seller might submit control agents so that $N$-agents will bid higher. Of course, if control agents are present, it is no longer an $N$-agent's best strategy to bid the user's valuation times $(N - 1)/N$.
- The *discover agent* computes the expected gain from bidding a small amount more than the current highest bid according to the agent's current distribution of the user's valuation. This is intended for settings where the user has a probability distribution over the item rather than an exact valuation. In the future, the probability distribution could be updated by new events, or by what others have bid in nonprivate value auctions. Such updating is part of our current research.

After choosing an agent type, the user customizes the agent by setting parameter values, such as user identification number, password, e-mail address to be used for reporting, agent name for agent management, and the user's reservation price for the desired item. When the user clicks the create button, the Java agent is automatically generated according to these parameters. It then travels to the agent dock, docks there, and bids at eAuctionHouse.

Besides ease of programming, these template agents can help put novice bidders on an equal footing with expert bidders. For some auction types, the optimal bidding strategy can be game-theoretically determined in advance. The user then simply enters preferences for the template agent, and the agent acts strategically on the user's behalf.

Figure 2 shows the steps in creating a mobile agent without programming. On the left is the first screen that appears when the user asks to create an agent to bid in an auction. For the given auction type, the system recommends three of the five agent types. Therefore, the radio buttons of the other two agent choices are not clickable for this particular auction type. The user in this example chooses to create an $N$-agent. The screen on the right shows the next step in creating an $N$-agent. The user specifies the parameters: user identification number, password, email address used by the mobile agent for reporting, agent name for agent management, and the user's reservation price for the item. When the user clicks the create button, the Java agent is automatically generated based on these parameters, travels to the agent dock (ecommerce.cs.wustl.edu in the example), docks there, and bids at eAuctionHouse.

**The search is not only focused on finding a partner, but on finding a desirable coalition structure.**

## AUTOMATED COALITION FORMATION

Our current research focuses on mobile agents for automated coalition formation in electronic auctions.[8,9] Economic efficiency can sometimes be improved if bidders form coalitions. Consider, for instance, an auction in which one seller is selling one item. One buyer wants part of the item and another buyer wants the remaining part. The sum of the amounts they are willing to pay separately exceeds the highest price offered for the whole item by other bidders. By forming a coalition, the two buyers and the seller benefit.

There are two main barriers for users across the Internet to form coalitions while bidding online. First, finding partners can be time-consuming. Second, bidders do not necessarily trust each other without a binding contract. Issues arising in coalition formation include who is in charge of bidding, what happens if some bidders refuse to pay after the coalition's bid wins, and how much each participant has to pay if the coalition wins.

To support automated coalition formation, we propose using mobile agents. With an appropriate communication mechanism, it can be easier for an agent to locate potential partners than it is for a person sitting at a computer to do so. For example, the agent can search in a public place where bidders or agents looking to form a coalition post partial bids in the hope of being combined with others. The search is not only focused on finding a partner, but more generally, finding a desirable coalition structure—that is, partitioning the agents into coalitions. This is a computationally hard search problem even if all the agents are in one location and their characteristics are commonly known.[9–11] Agents usually search orders of magnitude faster than humans. Furthermore, agents' time is less costly.

To solve the trust problem, a third-party site might be necessary. At the third-party site agents could sign binding contracts and check users' credit histories and reputations.

Although collusion can improve economic efficiency for buyers and sellers, it also involves speculation costs and can cause revenue loss for the auctioneer. For example, bidders can coordinate to keep their bids artificially low so as to get the item at a lower price than they otherwise would. However, considering the number and diversity of users in most Internet auctions, it seems unlikely that bidders across the Internet would be able to establish such a coalition. Therefore, automated coalition formation in the Internet auction setting may contribute more to the positive aspects of coalition formation than to the negative.

## FUTURE WORK

Future research includes developing additional prototype agents based on new game-theoretic analyses. We are also developing learning methods for updating the user's valuation distribution in settings where the user does not know the exact value of the auctioned item. The discover agent then uses that distribution to bid optimally on the user's behalf. Finally, automated coalition formation introduces new challenging problems to electronic auctions. These will be further studied in the continuing development of eAuctionHouse and Nomad. ■

### ACKNOWLEDGMENTS

An earlier version of this article appeared as "Mobile Agents in an Electronic Auction House," *Proc. Third Int'l Conf. on Autonomous Agents (AGENTS), Mobile Agents in the Context of Competition and Cooperation (MAC3) Workshop*, 1999, pp. 24–33.

This work is supported by the U.S. National Science Foundation under Career Award IRI-9703122 and grants IRI-9610122 and IIS-9800994. The authors would like to thank Kate Larson for her comments on drafts of this article.

### REFERENCES

1. J.A. Rodriguez-Aguilar et al., "A Java-Based Electronic Auction House," *Proc. Second Int'l Conf. Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 97)*, 1997.
2. T.W. Sandholm, "An Algorithm for Optimal Winner Determination in Combinatorial Auctions," *Proc. 16th Int'l Joint Conf. Artificial Intelligence (IJCAI)*, Morgan Kaufmann, San Francisco, Calif., 1999, pp. 542–547.
3. S.J. Rassenti, V.L. Smith, and RL. Bulfin, "A Combinatorial Auction Mechanism for Airport Time Slot Allocation," *Bell J. of Economics*, Vol. 13, 1982, pp. 402–417.
4. M.H. Rothkopf, A. Pekec, and R.M. Harstad, "Computationally Manageable Combinatorial Auctions," *Management Science*, Vol. 44, No. 8, 1998, pp. 1131–1147.
5. T.W. Sandholm, "eMediator: A Next-Generation Electronic Commerce Server," to appear in *Proc. Fourth Int'l Conf. Autonomous Agents (AGENTS)*, June 2000.
6. D. Wong et al., "Concordia: An Infrastructure for Collaborating Mobile Agents," *First Int'l Workshop on Mobile Agents (MA)*, Springer Lecture Notes in Computer Science 1219, Berlin, Apr. 1997.
7. E. Rasmusen, *Games and Information*, Basil Blackwell Press, 1989.
8. T.W. Sandholm and V.R. Lesser, "Coalitions Among Computationally Bounded Agents," *Artificial Intelligence*, Vol. 94, No. 1, 1997, pp. 99–137.
9. T.W. Sandholm et al., "Coalition Structure Generation with Worst-Case Guarantees," *Artificial Intelligence*, Vol. 111, Nos. 1–2, 1999, pp. 209–238.
10. O. Shehory and S. Kraus, "A Kernel-Oriented Model for Coalition-Formation in General Environments: Implementation and Results," *Proc. Nat'l Conf. on Artificial Intelligence (AAAI)*, AAAI Press, Menlo Park, Calif., Aug. 1996, pp. 134–140.
11. O. Shehory and S. Kraus, "Methods for Task Allocation via Agent Coalition Formation," *Artificial Intelligence*, Vol. 101, Nos. 1–2, 1998, pp. 165–200.

Tuomas Sandholm is assistant professor of computer science at Washington University, St. Louis. He received the PhD and MS degrees in computer science from the University of Massachusetts at Amherst, and an MS in Industrial Engineering and Management Science from the Helsinki University of Technology, Finland. His research interests include artificial intelligence, ecommerce, game theory, multiagent systems, auctions, automated negotiation and contracting, coalition formation, and normative models of bounded rationality. He has published over 100 technical papers and received several academic awards including the NSF CAREER award. (http://www.cs.wustl.edu/~sandholm)

Qianbo Huai is an engineer at Microsoft. He received an MS degree from Washington University, St. Louis, where he was involved in Sandholm's Multiagent Systems Research Group.

Readers can contact Sandholm at sandholm@cs.wustl.edu and Huai at qhuai@microsoft.com.