MultiHammer: A Virtual Auction System based on Information Agents

Ryota Yamada[a], Hiromitsu Hattori[a], Takayuki Ito[b], Tadachika Ozono[a], and Toramatsu Shintani[a]

[a] *Dept. of Intelligence and Computer Science, Nagoya Institute of Technology*
Gokiso, Showa-ku, Nagoya, 466-8555, JAPAN.
Tel: +81-52-744-3153, Fax: +81-52-735-5584, E-mail: {ryota,hatto,ozono,tora}@ics.nitech.ac.jp

[b] *Center for Knowledge Science, Japan Advanced Institute of Science and Technology*
1-1 Asahidai, Tatsunokuchi-machi, Nomi-gun, Ishikawa, 923-1292, JAPAN.
Tel: +81-761-51-1813, Fax: +81-761-51-1149, E-mail: itota@jaist.ac.jp

Abstract

In this paper, we propose a virtual auction system based on information agents. We call the system the MultiHammer. MultiHammer can be used for studying and analyzing online auctions. MultiHammer provides functions for implementing a meta online auction site and an experiment environment. We have been using MultiHammer as an experiment environment for BiddingBot. BiddingBot aims at assisting users to bid simultaneously in multiple online auctions. In order to analyze the behavior of BiddingBot, we need to purchase a lot of items. It is hard for us to prepare a lot of fund to show usability and advantage of BiddingBot. MultiHammer enables us to effectively analyze the behavior of BiddingBot. MultiHammer consists of three types of agents for information collecting, data storing, and auctioning. Agents for information collecting work specifically at their auction site as flexible wrappers. To make agents work as wrappers, we need to realize software modules for each online auction site. Implementing these modules requires a lot of time and patience. To address this problem, we designed a support mechanism for developing the modules. Agents for data storing record the data gathered by agents for information collecting. Agents for auctioning provide online services using data recorded by agents for data storing. By recording the activities in auction sites, MultiHammer can recreate any situation and trace auction for experimentation. Users can participate in virtual auctions using the same information in real online auctions. Users also can participate in real auctions via wrapper agents for information collecting.

Keyword:
Online Auctions; Information Collection; Web Wrappers; Multi Agent System

1 Introduction

Since its inception, the Internet has grown at a staggering rate. Increasingly, there has been a growing interest in online auctions as a means of *electronic commerce*. Particularly agent-mediated *electronic commerce* for users attracts attention[1]. One of the reasons for this interest is that agents can autonomously and cooperatively act on the behalf of their users on a network environment. On the Internet, there exist a lot of online auction sites: eBay[2], Yahoo! Auctions[3], Amazon.com Auctions[4], and so on. At each online auction site, a lot of people trade items asynchronously. However, users do not always know where the best auction site is, how to participate in them, where they should buy or sell their items, and so on. To address these problems, we built a virtual auction system that can help users finding items, studying the means of participating in and analyzing online auctions, among other things.

There is a lot of related work in the field of *electronic commerce*. Agent-mediated *electronic commerce* systems can be classified into two categories. One category is shopping support systems, which support users in searching for and gathering information on desired items. In this category, there are such systems as iTrack[5], AuctionWatch[6], DealTime[7], ShopBot[8], and Jango[9]. The other is virtual market systems, which provide a place in which agents or users can conduct auctions or other types of *electronic commerce*. AuctionBot[10], Kasbah[11], FishMarket[12], Teta-a-Teta[13], and eMediator[14] can be classified in this category.

In this paper, we propose a virtual auction system that employs multi-agent technologies. We call the system the *MultiHammer*. The difference between *MultiHammer* and the existing shopping support systems is that while the existing systems can extract information on shopping over the Internet, only *MultiHammer* can provide services using extracted information for users to help them study or analyze real online auctions. In existing virtual market systems, agents can negotiate in auctions or other forms of *electronic commerce* that are provided by the system used, but cannot use real online auction sites. However, using *MultiHammer*, users can participate in them.

This paper consists of five sections. In Section 2, we show

the outline of *MultiHammer* that is based on multi-agents for studying, analyzing, and participating in practical online auction sites. In Section 3, we propose information collecting mechanisms using multi-agents for multiple auction sites. In Section 4, we discuss examples of applying *MultiHammer* and the features of *MultiHammer*. Finally, in Section 5, we present our concluding remarks and discuss our future work.

2 *MultiHammer*: A Virtual Auction System based on Information Agents

2.1 The Outline of *MultiHammer*

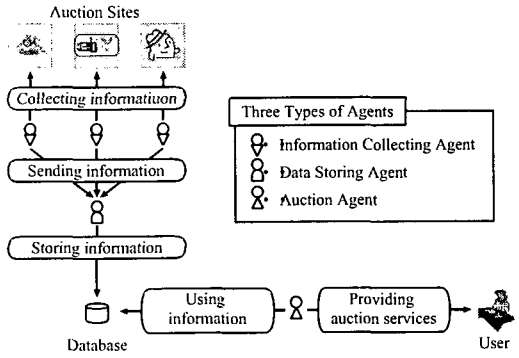


Figure 1. The System Architecture of MultiHammer

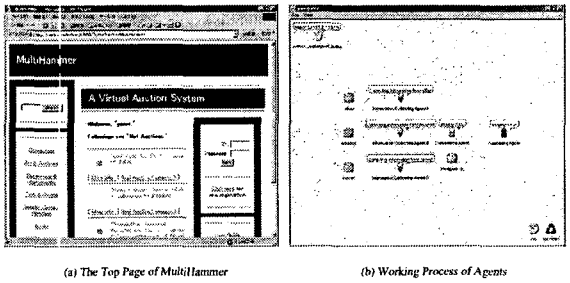
Figure 1 shows the architecture of *MultiHammer*. *MultiHammer* consists of an auction agent, a data storing agent, and several information collecting agents, each of whom is assigned to an auction site. The information collecting agents cooperatively and simultaneously gather information. The data storing agent stores the information to a database. The auction agent provides online auction services using the stored information. By recording the activities in auction sites, *MultiHammer* can recreate any situation and trace auction for experimentation.

In *MultiHammer*, each of the information collecting agents works specifically at its auction site and can behave as a flexible wrapper[15], because different sites present information in different forms. In this paper, we focus on the gathering mechanism of information collecting agents. These agents work as wrappers not only for collecting information, but also for sending information(e.g., querying or bidding for items). Using these abilities, users can search for items and bid on them using auction agents.

2.2 Implementation based on MiLog

MultiHammer was implemented using MiLog[16], Java[17], and Ruby[18]. The information agents were realized by using MiLog (a multi-agent development language), which helps users realize a multi-agent web server using Prolog programming and Java programming. MiLog was implemented in Java. MiLog agents can be extended by using Java.

The data storing agents, which are realized by using MiLog, were added the ability to access a database using JDBC (an API that allows access to virtually any tabular data source in the Java programming language)[19]. The auction agent was realized by using Ruby (an interpreted scripting language for quick and easy object-oriented programming) as a CGI program. The left side of Figure 2 shows the top page of the interface for *MultiHammer*. The right side of Figure 2 shows the interface for monitoring the working process of the information collecting agents and the data storing agent. We call this monitoring tool the *Agent Monitor*. *Agent Monitor* was implemented by using MiLog.



(a) The Top Page of MultiHammer (b) Working Process of Agents

Figure 2. The User Interface of MultiHammer

3 Information Collecting Mechanisms

In most cases, access to information in online auction sites is granted through a Web gateway with forms as a query language and HTML as a display vehicle. While this architecture offers convenient access to human users, it is not suitable for agents. An unfortunate problem with online auction sites is that they exist independently. For instance, when you want to compare the price of items in different online auction sites, you have to access each of the sites, search for items, get prices, and compare. All of these have to be executed by clicking, fetching, and reading Web documents. It is a rather tedious process. The aim of *MultiHammer* is to automate the entire process. In order to automate the process, we need Web wrappers. A wrapper is a software component that converts data and queries from one form to another. In the Web environment, a wrapper should convert information implicitly that is stored as an HTML document into information explicitly stored as a data-structure, which is then processed further. The role of a wrapper is actually three-fold: (1) retrieving a Web document, (2) extracting information from the document, and (3) exporting the extracted information in a structured way[20]. Moreover, a wrapper should cope with the unstable nature of the Web environment; a wrapper should effectively handle network failures, ill-formed documents, changes in the layout, and other problems.

In *MultiHammer*, information collecting agents work as wrappers for each site. The agents can retrieve Web documents for items in online auction sites, via HTTP, using built-in predicates (commands) of MiLog. To extract information from Web documents, information collecting agents

need data on their structure. To export information in a structured way, agents need data on the semantics of the extracted information. This data should be specified on each online auction site. We prepared this data for the information collecting agents as plug-in modules that are implemented based on MiLog. We developed these modules for three online auction sites: eBay, Yahoo! Auctions, and Amazon.com Auctions.

In order to develop plug-in modules, we must analyze HTML descriptions of Web documents. This process requires a lot of time and patience. To address this problem, we designed a mechanism to support the development of the modules to extract information on items in an online auction site. Figure 3 shows our plug-in modules and their development support process. The development support process consists of the following five steps: (i) Retrieving two Web documents on different items in the same online auction site and convert them in each of their tree structures. (ii) Comparing each node in the two tree structures and finding templates for the Web documents on the items in their respective sites. (iii) Using the templates to extract information on an item from a Web document. (iv) Structuralizing the extracted information on an item using data on the Web document. Currently, the order of the information given in the Web documents is used for constructing a structure. (v) Checking the structured information. If necessary, the templates and data must be amended. Based on the above process, we can prepare plug-in modules without analyzing the HTML description in detail.

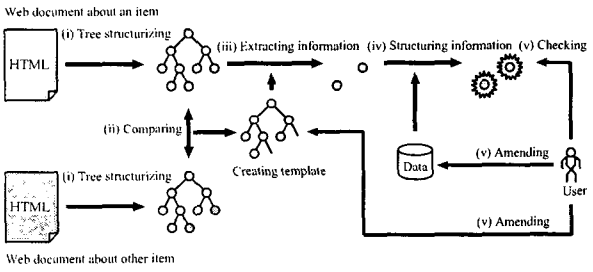


Figure 3. Developing Plug-in Modules

4 Discussion

In this section, we discuss examples of applying *MultiHammer* and the features of *MultiHammer*. In Section 4.1, we show four examples of applying *MultiHammer*. In Section 4.2, we present three features of *MultiHammer*.

4.1 Examples of Applications

A Testbed for Online Auctions: In *MultiHammer*, agents provide online services that simulate real sites. In real auction sites, a lot of services for participants are available: user registration, item searches, proxy bidding, winner determination, and so on. Beginners at online auctions often get confused and intimidated by the vast number of services, and retreat from conducting deals. Using *MultiHammer*, beginners can try various services freely without any money and will begin to enjoy online auctions.

An Analyzing Tool for Online Auctions: In *MultiHammer*, the data storing agent stores all of the information collected by the information collecting agents. Using the stored information, participants can analyze auctions. In real online auctions, similar items tend to be marked down to similar prices. According to this heuristics, participants find it valuable to know successful bid results on similar items so that they can predict the marked-down price. In *MultiHammer*, participants can analyze auctions by using the data on past ones.

A Meta Online Auction Site: Information collecting agents work in *MultiHammer* as wrappers for each of their online auction sites. Using these agents as wrappers for online auction sites, users can access many online auction sites as if they were accessing just one. Using *MultiHammer*, users can participate in many sites without knowing the details of each site.

An Experimental Environment for Intelligent Electronic Commerce Support System: While recording the history of all of the activities in the auction sites, *MultiHammer* can restore any situation and trace auction for experimentation. We developed *BiddingBot*[21] to support users of real auction sites. In real auction sites, many people trade simultaneously. Even if we participate in auctions in real auction sites using *BiddingBot* only for experimental purposes, we are bound by the contracts. In order to analyze the behavior of *BiddingBot*, we need to purchase a lot of items. It is hard for us to prepare a lot of fund to show usability and advantage of *BiddingBot*. In addition, we can not meet the same situations for experiments in real sites. Namely, the real sites are not suitable for experimentation. *MultiHammer* provides an experimental environment for the *BiddingBot* where researchers can test their programs for real auctions. *MultiHammer* enables us to effectively analyze the behavior of *BiddingBot*. Generally, if someone tries to build a system that can actually make a deal in *electronic commerce* sites, he needs a kind of virtual *electronic commerce* system we proposed.

4.2 The Features of *MultiHammer*

Asynchronous Information Collecting from Multiple Auction Sites: In *MultiHammer*, agents for information collecting work as wrappers for Web documents in each of their auction site. The agents collect information from each of their auction site asynchronously. Using the agents, *MultiHammer* can gather information in several auction sites. In order to work as wrappers for Web documents, the agents need plug-in modules for information collection. We can prepare the modules for the agents using our plug-in modules developing support process proposed in Section 3.

Supporting Bidding in Multiple Auction Sites: In *MultiHammer*, agents for information collecting can also work as wrappers for online services in each of their auction site. The agents send query from users to each of their auction site. Using the agents, *MultiHammer* can provide a common interface for several auction sites. In order to work as wrappers for online services, the agents need plug-in modules for sending queries. Analyzing each of auction site, we developed the modules for the agents by hand.

Restoring Situations and Tracing Auctions: In *MultiHammer*, agents collect information from several existing auction sites. Storing the collected information with time stamp, *MultiHammer* can record the histories of auctions in several auction sites. Referring the histories of auctions, *MultiHammer* can restore situations and trace auctions.

5 Conclusion and Future Work

In this paper, we proposed a virtual auction system based on information agents. We call the system the *MultiHammer*. *MultiHammer* consists of three types of agents for information collecting, data storing, and auctioning. Agents for information collecting work specifically at their auction site as flexible wrappers. To make agents for information collecting work as wrappers, we need to realize software modules for each online auction site. Implementing these modules requires a lot of time and patience. To address this problem, we designed a support mechanism for developing the modules. Agents for data storing record the data gathered by agents for information collecting. Agents for auctioning provide online services using data recorded by agents for data storing. By recording the activities in auction sites, *MultiHammer* can recreate any situation and trace auction for experimentation. Users can participate in virtual auctions using the same information in real online auctions. Users also can participate in real auctions via wrapper agents for information collecting. *MultiHammer* can be used as a testbed and analyzing tool for online auctions. *MultiHammer* provides functions for implementing a meta online auction site and an experiment environment. We have been using *MultiHammer* as an experiment environment for *BiddingBot*. *BiddingBot* aims at assisting users to bid simultaneously in multiple online auctions. Generally, if someone tries to build a system that can actually make a deal in *electronic commerce* sites, such as the *BiddingBot*, he needs a kind of virtual *electronic commerce* system we proposed.

Currently, we are preparing plug-in modules for information collecting agents using a support process. Ideally, the entire process for developing the modules should be automated. In order to automate the process, we need to improve the support process or create other methods. In *MultiHammer*, users can participate in auctions using the same data as real sites. The changes in these sites affect *MultiHammer*. However, the changes in *MultiHammer* affect real sites only when users want to participate in real auctions. This is one of the convenient and powerful aspects of *MultiHammer*. However, this can be a problem when we use *MultiHammer* as an experimental environment. As the bids in *MultiHammer* do not affect real auctions, the participants of these auctions do not raise their bids. To address this problem, we should implement agents that behave as counter bidders in *MultiHammer*. Implementing agents that behave as counter bidders, *MultiHammer* can provide a more realistic and stimulating experimental environment for users.

References

[1] Guttman, R. H.; Moukas, A. G.; and Maes, P. 1998. Agent-mediated Electronic Commerce: A Survey. *The Knowledge Engineering Review* 13(2):147–159.

[2] eBay. `http://www.ebay.com/`.

[3] Yahoo! Auctions. `http://auctions.yahoo.com/`.

[4] Amazon.com Auctions. `http://auctions.amazon.com/`.

[5] iTrack. `http://www.itrack.com/`.

[6] AuctionWatch.com. `http://www.auctionwatch.com/`.

[7] DealTime.com. `http://www.dealtime.com/`.

[8] Doorenbos, R. B.; Etzioni, O.; and Weld, D. S. 1997. A Scalable Comparison-Shopping Agent for the World-Wide Web. In Proceedings of the First International Conference on Autonomous Agents, 39–48. Marina del Rey: ACM Press.

[9] Jango. `http://www.jango.com/`.

[10] Wurman, P. R.; Wellman, M. P.; and Walsh, W. E. 1998. The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents. In Proceedings of the Second International Conference on Autonomous Agents, 301–308. Minneapolis: ACM Press.

[11] Chavez, A.; and Maes, P. 1996. Kasbah: An Agent Marketplace for Buying and Selling Goods. In Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, 75–90. London: The Practical Application Company.

[12] Rodriguez, J. A.; Noriega, P.; Sierra, C.;and Padget, J. 1997. FM96.5 A Java-based Electronic Auction House. In Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, 207–224. London: The Practical Application Company.

[13] Guttman, R. H.; and Maes, P. 1998. Agent-mediated Integrative Negotiation for Retail Electronic Commerce In Proceedings of the Workshop on Agent Mediated Electronic Trading, 77–90. Minneapolis: ACM Press.

[14] Sandholm, T. W. 1999. eMediator: A Next Generation Electronic Commerce Server. In Proceedings of the Sixteenth National Conference on Artificial Intelligence, 923–924. Orlando: AAAI Press.

[15] Wells, D. 1996. Wrappers. `http://www.objs.com/survey/wrap.htm`.

[16] Fukuta, N.; Ito, T.; and Shintani, T. 2000. MiLog: A Mobile Agent Framework for Implementing Intelligent Information Agents with Logic Programming. In Proceedings of the First Pacific Rim International Workshop on Intelligent Information Agents (PRIIA2000), 113–123. Melbourne: Deakin University.

[17] Arnold, K.; Gosling, J.; and Holmes, D. 2000. The Java Programming Language, Third Edition. : Addison-Wesley.

[18] Ruby Home Page. `http://helium.ruby-lang.org/en/`.

[19] White, S.; Fisher, M.; Cattell, R.: Hamilton, G.; and Hapner, M. 1999. JDBC API Tutorial and Reference, Second Edition: Universal Data Access for the Java2 Platform. : Addison-Wesley,

[20] Sahuguet, A.; and Azavant, F. 1998. Wysiwyg Web Wrapper Factory (W4F). `http://db.cis.upenn.edu/DL/WWW8/`.

[21] Ito, T.; Fukuta, N.; Shintani, T.; and Sycara, K. 2000. BiddingBot: A Multiagent Support System for Cooperative Bidding in Multiple Auctions. In Proceedings of the Fourth International Conference on Multi Agent Systems, 399–400. Boston: IEEE Computer Society.