Crossroads
The ACM Student Magazine

RAS: A System for Supporting Research in Online Auctions

By ***Jarrod Trevathan*** and ***Wayne Read***

Introduction

Online auctioning is unparalleled as the fastest growing exchange medium to emerge from electronic commerce technology. Buyers and sellers located around the world now auction various items from the latest DVD to rare collectables. eBay [**2**] and uBid [**10**] are among the most successful and popular of the commercial online auctioneers. They use an auctioning process based on a type of auction referred to as an *English* auction. In an English auction, bidders outbid each other for an item. The winner is the bidder with the highest bid.

Despite the popularity of auctioning online, there are inherent security risks. For example, bidders might not pay or sellers might not deliver the item. To compound the problem, commercial auctioneers do not hold themselves liable for any transactions that go awry. Cryptographic and other security methods can help protect against many of these risks. However, there is no way to test the feasibility of such auction mechanisms. Therefore, an evaluation environment for research purposes is required.

In 1998, Wellman et al. [**11**] developed an online auction server called the AuctionBot. Designed at the University of Michigan, the AuctionBot's main purpose was to serve as an infrastructure for conducting auction research. The AuctionBot strived to be *generic* by its capacity to run various types of auctions (i.e., English, Vickrey, and so on). Specific types of auctions could be performed by altering the AuctionBot's parameters. The AuctionBot also contained an agent interface for software bidding agents.

Software bidding agents bid on behalf of a human bidder according to a predetermined strategy. After the AuctionBot was decommissioned in 1999, its platform became the basis for the 2001 Trading Agent competition [**14**]. The competition pitted bidding agents against each other. The agents participate in an elaborate economic game and are assessed on their ability to acquire certain resources. Other literature on the AuctionBot can be found in [**12**,**13**].

In 1999, Kumar and Feldman [**4**] presented a software model for auctioning online. The design was based on an auction system they had implemented. Similar to the AuctionBot, they also sought a generic auction architecture. Sound software engineering principles are applied by employing an object-oriented approach to auction design. Kumar and Feldman describe auction web site management, navigational issues and the process flow of an auction. In addition, they discuss the effect of timing and security issues with online auctions.

Despite these initial attempts at auction software design, publicly available research activity has since receded. This is largely due to the dominance of commercial online auctioneers. To compound the problem, commercial auctioneers tend not to publish their research.

We required our own infrastructure for performing research. There are many vendors selling auction software online. However, such software is expensive and cannot be customised for our research needs. Due to this fact and the limited availability of academic literature on auction software design, we have created our own online auction server.

This article describes our experiences with designing online auction software and presents a client-server software model for conducting online English auctions. The model is based on an existing auction server developed at James Cook University called the "*Research Auction Server*" (RAS). We use RAS to perform both simulated and real auctions and to gather data on the performance of key areas of auction research. RAS has been used to conduct research in auction security, fraudulent bidding behavior, bidding agent design, and market clearing algorithms. RAS is open source and available online at **http://auction.maths.jcu.edu.au**.

In this article we discuss the mechanics of English auctions, then we describe the major software components of an online auction, the processes involved, and web site

navigation. Next, we present an object model of the system along with a database schema and transaction issues. We address timing concerns and specify a software bidding agent interface. Finally, we discuss research conducted on RAS and its implications for transaction settlement.

Online English Auctions

There are many types of auctions such as English, Vickrey, Dutch, double, an so on. (See Trevathan et al. [**8**]). The *English* auction is the most well known type of auction. Formally, an English auction is referred to as an *ascending price* auction. To win, bidders must outbid each other, forcing the price up. The winner is the bidder with the highest price at the end of the auction.

The form of a bid in an English auction is:

< bidder, price, time >

At any time in an online auction, a bidder can request a *price quote* from the auctioneer. The price quote contains the bid information of the current highest bid.

The *bid history* of an auction is a list of all bids that have been submitted. Table 1 gives an example bid history for an auction. Here the winning bid is $55.

Table 1 - Bidding History

Bid	Bidder	Price	Time
7	Wayne	$ 55	33:05
6	Jarrod	$ 50	30:15
5	Sharith	$ 40	24:22
4	Wayne	$ 35	17:01
3	Mr Gab	$ 25	12:35
2	Shane	$ 20	05:44
1	Jarrod	$ 15	02:09

English auctions can have three additional parameters:

- *Starting price* - the minimum price at which the bidding must commence.
- *Reserve price* - the minimum price the seller of the item will accept. If the final price is below the reserve, the auction result is void.
- Minimum *bid increment* - the minimal amount required to outbid the current highest offer.

Online auctions allow many variations on English auctions. For example, bidders can alter or cancel bids, auction multiple goods, and issue sell bids. We will not deal with all of the possible extensions to English auctions and restrict our attention to the essential features.

Previous auction software architectures have strived to implement as many auction types as possible. However, the model in this article only focuses on English auctions. There are several reasons for this. First of all, English auctions are the most common type of auction performed online. Secondly, there is limited literature on software design for online English auctions. Furthermore, some auctions (for example, continuous double auctions) are more complicated than English auctions. Therefore our auction software model allows a sufficient amount of detail.

Components of an Online Auction

Figure 1 presents a high level software model for performing online English auctions. There are two main parties: a bidder and an auctioneer. The parties are joined by a communication link.

There are two types of interfaces for a bidder. The first is the web interface. This is for a human bidder. The bidder interacts with the auctioneer via an HTML browser. The second interface is for a software bidding agent. A bidding agent interacts with the auctioneer using an application programming interface.

The auctioneer runs a web server (e.g., Apache) and a scripting language, in this case php. The entire auction is database driven. All state information (e.g., bids and timing) about the auction is contained in the database. When a client submits a bid or requests a price quote, a database transaction occurs. The database generates dynamic web pages in response to bidder activity using the scripting language.

Processes Involved

There are several main activities in an online English auction:

- *Initialization*: The auctioneer sets up the auction and advertises it, i.e., type of good, starting time, etc.
- *Registration*: In order to participate in the auction, bidders must first register with the auctioneer.
- *Price quote*: A bidder obtains a price quote from the auctioneer.
- *Bidding*: A bidder submits a bid to the auctioneer.
- *Winner Determination*: The auctioneer determines the winner according to the auction rules.

Transaction settlement/payment is the process of collecting payment from and delivering the goods to the winning bidder.

Bidder/Winner Notification refers to the processes involved with informing a bidder with information other than the price quote. This is information specific to an individual such as confirmation of bid receipt or notifying the winner that they have won.

Web Interface Navigation

Figure 2 illustrates how human bidders navigate the auction website. Each bubble represents a major section of the site. Lines represent links between sections and arrows indicate the navigational direction a user must follow to reach a particular page.

After registering, a user can log in using a password. Upon login a user enters the secure area (shown by the red box in Figure 2). This is a collection of pages which contains operations that only a registered user can perform.

The home page is the main area for the bidder. Bidders are able to search for a listed auction. Once a bidder has selected an auction, they are able to obtain a description of the item and information regarding the auction (i.e., price quote, minimum increment, and time remaining). The bidder can then use this information to submit a new bid. After submitting a bid, a bidder can return to the home page, search for a new auction, or return to the current auction.

In the secure area, specific information regarding the user (such as user id and password) must be carried from one page to another. To achieve this, the secure area is implemented using session variables. Session variables are similar to cookies, as they are used to store information for a particular period of time. The values in session variables exist until the session terminates. This allows information to be passed between web pages or to another web site. A session can be created using a session identifier which is stored on the server. When the client makes a request, the data stored in the session variable can be accessed any number of times until the session ends.

An alternative approach for passing information between pages is to use a query string. However, using a query string in this manner is difficult and cumbersome. Session variables on the other hand allow information to be more easily retrieved and maintained. Figure 3 illustrates the process involved for using session variables.

When a bidder logs off, the session variable is destroyed. The variable is also destroyed after a predetermined amount of bidder inactivity (for example, when a user forgets to logoff).

Upon submitting a bid, the bidder receives an email confirming that his/her order has been received. The bidder will also be informed of such information via the web page.

Search is an important mechanism in online auctions. Bidders have the ability to choose from millions of auctions to participate in. The search mechanism must allow a bidder to quickly and accurately identify a desired auction. Items for auction are typically categorised hierarchically. For example, the category "automobiles" would contain auctions for cars, motorbikes, trucks, etc. The "car" category would be further decomposed into auctions for sedans, hatchbacks, station wagons, etc. Other items for auction might be listed according to the seller. A full description of searching methods is beyond the scope of this article, however, we leave the architecture of RAS open to incorporate any search mechanism.

Object Model

The web interface for RAS allows human bidders to interact with the auctioneer. Figure 4 depicts an object model for RAS's web interface. There are two objects: *Bidder* and

Auctioneer. The top portion of each object indicates its internal state and the bottom portion lists its methods (see Coad and Yourdon [**1**]).

A *Bidder*'s internal state is the *userId* and *password*. The bidder interacts with the auctioneer via the auctioneer's methods. The bidder only has two methods: *register* and *determine bid*. During registration, a bidder obtains a userId and password, which are kept secret. When participating in an auction, a human bidder determines his/her bid according to individual preferences.

The *Auctioneer* manages the database, which is the central component of the auction. This contains all information regarding bidders, their bids, and the auctions conducted. There are two types of methods the auctioneer can perform: private and public. Private methods are operations that only the auctioneer can perform, such as list a new auction (*list auction*), *register* new bidders, and *terminate* an auction. Public methods are services which are requested by a bidder. This includes the ability to *login*, *search* for an auction, request the minimum *increment*, obtain a *price quote*, request the *time remaining*, submit a bid (*submit bid*), and *logoff*.

Database

Figure 5 depicts an entity-relationship diagram for the database in an online English auction. Rectangles represent entities, diamonds represent relationships, and ellipses represent attributes (see Silberschatz et al. [**5**]). There are three entities: *User*, *Auction*, and *Bid*. The primary key for each entity is underlined.

When a bidder registers, their details are entered into the user entity. Each user is identified by a unique *userId*. Information stored about a user includes their *password*, *name*, and *email* address.

Each auction is identified by a unique *auctionId*. Information stored about an auction includes an item *description*, *start* time, *expiration* time, *startPrice*, *reserve* price, minimal *increment*, and *notes* about the item. The *status* attribute indicates whether

the auction is currently active or has terminated.

Every time a bidder submits a bid, all information about the bid is entered into the bid entity. Each bid is identified by a unique *bidId*. To associate a bid with a bidder, the bid entity stores the *userId* as a foreign key. To associate a bid with an auction, the bid entity also stores the *auctionId* as a foreign key. Other bid information includes the *price* of a bid and a *timestamp* that indicates when the bid was submitted.

In this model a user cannot list items in an auction directly. Instead this must be done exclusively by the auctioneer. Allowing a bidder to list items for sale requires an extension to the entity-relationship diagram. The auction essentially becomes a double auction (i.e., many buyers and many sellers) which complicates matters.

There are several database transactions that occur during an auction:

- Initialization - A new auction's details are entered into the database.
- Registration - A new user's details are entered into the database.
- Login - An existing user's details are retrieved from the database.
- Price Quote - An existing auction's details and bid history are retrieved from the database.
- Bid Submission - A new bid's details are entered into the database.
- Auction Termination/Winner Determination - An existing auction's details and bid history are retrieved from the database. The auction's status is set to 'inactive' and entered in the database. The winner is determined according to the auction's rules.

Timing

When a bid is received for a particular auction, it is compared to the current highest bid. If it does not meet the minimal amount required to outbid the existing highest bid, then it is discarded. A bid that does meet the required amount is timestamped by the auctioneer and entered into the database.

Online English auctions can terminate according to the following rules (see [**4**, **6**]):

- Expiration Time - The auction closes at a predetermined expiration time.
- Timeout - The auction closes when no bids higher than the current highest bid

are made within a predetermined timeout interval.
- Combination of Expiration and Timeout - The auction closes when there is a timeout after the expiration time.

In our model, the auction is database driven. Rather than having a running program continually check the time in order to terminate the auction, our auction terminates in response to bidder activity. This means that every time a bidder submits a bid or requests a price quote, a script is run to check whether the auction has terminated according to the database clock. This involves comparing the database time to the expiration attribute of the auction entity in the auction entity-relationship diagram (Figure 5). A pseudocode script for terminating an auction with an expiration time is shown below (in blue):

```
if database time >= expiration {
    set auction status to inactive
    select highest bid from auction
    if bid > reserve
        declare this bid the winner
    else
        auction is void
}

else {
    expiration = database time + timeout
}
```

This setup does not require synchronization between the bidders and the auctioneer.

Terminating an auction using a timeout involves adding additional time to the auction's expiration attribute in the database every time a higher bid is received. Let *timeout* be a constant time increment. Adding a timeout extends the terminate auction script as shown by the red section of the code.

Some variants of English auctions allow bid cancellation. This is common in English auctions which terminate using an expiration time. The justification for this is that such auctions can often take days or weeks. In this situation, a bidder may be reluctant to make such an open-ended bid.

The simplest approach to bid cancellation is to delete the cancelled bid from the database. However, for security reasons a cancelled bid should be recorded. This is achieved by including *valid* and *cancellationTime* attributes in the bid entity of the entity-relationship diagram shown in Figure 5. Figure 6 illustrates new additions to the bid entity where the red indicates the new attributes.

A bid is initially in a valid state and the cancellation time is *null*. This indicates that it is to be considered for an auction. If a bid is cancelled, it enters an invalid state, indicating that it should not be considered for an auction. This allows the bid details and time of cancellation to be recorded but not included in an auction.

Bidding Agents

A bidding agent is a program which bids on behalf of a human bidder. In terms of an English auction, a bidding agent is permitted to outbid any bid until the bidding price exceeds a maximum amount specified by the human bidder. In auctions that can last days or weeks, bidding agents remove the need for a bidder to constantly observe an auction. Instead, a bidding agent monitors the auction proceedings for any price activity and responds in accordance with its programmed strategy.

The object model for a simple bidding agent is shown in Figure 7. The auctioneer must provide an agent with six basic services: *login*, request *price quote*, minimum *increment*, *time remaining*, *submit bid* and *logoff*. We have not listed the auctioneer's private methods in the model.

Similar to a human bidder, an *Agent* object has a *userId* and *password*. An agent also knows the *auctionId* of the auction it wants to participate in. The *valuation* is the maximum limit that an agent can bid. An agent can perform two operations: *initialize* and *determine bid*. The initialize method provides the agent with the human's userId and password, and instructs the agent which auction to participate in and the valuation of the auctioned goods.

The following pseudocode script of the *determine bid* method illustrates the basic operation of a bidding agent. Green code indicates the methods of the Auctioneer object that are used by the agent.

```
bid = 0
login ( userId, password, auctionId )
repeat {
    if ( bid <= price quote ) {
        bid = price quote + increment
        if ( bid <= valuation AND time remaining )
            submit bid ( bid )
        else
            logoff
    }
}
```

Initially the agent's bid is zero. The agent logs in by supplying its userId and password. The agent also indicates the auction it wants to participate in by specifying the auctionId. Once logged in, an agent repeatedly requests a price quote and alters its strategy accordingly.

In our simple agent example, the agent's current bid is compared to the current highest bid in the auction. If the agent's bid is less than the price quote, then the agent increases its bid by a given increment. If the new bid does not exceed the agent's preset valuation, then the agent submits the bid to the auctioneer. Otherwise the agent logs off as it is not allowed to exceed the valuation. Before submitting a bid, the agent also checks whether the auction has terminated. If so, the agent logs off.

Bidding agents are not limited to the strategy employed by our simple agent example. Bidding agents can employ sophisticated strategies. Some of these strategies are based on statistical analysis of the auction data and neural network reasoning about the best strategy to take.

Research conducted using RAS

This section briefly describes research we have been conducting on RAS.

Security and Anonymity

Security and anonymity are crucial in online auction software design. Unlike a traditional auction, the participants in an online auction are not physically present. This presents many opportunities for cheating. For example, a bidder might repudiate having made a bid, forge a bid on behalf of another bidder, or refuse to pay. Furthermore, the auctioneer might be corrupt and award the auction to someone other than the legitimate bidder. Additionally, outsiders may attempt to influence or disrupt the auction proceedings. Online auctioning also raises many concerns regarding the privacy of a bidder's personal information, e.g., identity and bidding history.

Security in auctions has been discussed in the literature (for example, see [**4**, **6**, **7**]). Moreover, various cryptographic auctioning schemes have been proposed. The main goals of such schemes are to prevent the forging of bids and bid repudiation, to protect against auctioneer corruption, and to preserve the anonymity of bidders. Cryptographic schemes attempt to achieve this by signing and encrypting bids, and by employing anonymity preserving mechanisms and publicly verifiable protocols.

A secure scheme must be able to provide security with a reasonable quality of service to auctions involving large numbers of bidders. Cryptographic operations such as bid signing/verification and encryption are computationally expensive, requiring time to perform. This effectively slows down the auctioning process and may result in bids not being included in an auction.

RAS is used as a platform to develop and test security mechanisms for online auctions. RAS has been used to implement existing cryptographic auction schemes in order to determine their feasibility by measuring signing/verification and encryption times. We have also proposed schemes of our own (see [**8**, **9**]), which we have implemented using RAS.

RAS is also used to develop mechanisms to detect and prevent fraudulent bidding practices such as shilling. Shilling is where the seller introduces fake bids to drive up the final price. This is a problem as it forces a legitimate bidder to pay more for an item. English auctions are particularly susceptible to shill bidding behavior. We have identified the core strategies of a shill bidder and are developing methods to detect shill bidding. Both real and simulated auctions are conducted, some of which involve fraudulent bidding behavior. This allows us to test the effectiveness of our fraud detection techniques and to observe the reactions of innocent bidders.

Software bidding agents are used to simulate auctions involving large numbers (i.e., thousands) of bidders.

Transaction Settlement/Payment Mechanism

The payment mechanism is used to settle the auction (i.e., collect payment from the winner and ensure delivery of the item bid upon). The method employed by RAS depends on whether the auction is simulated or real.

During simulated auctions, fake items are auctioned. Bidders are issued a fake amount of money to use. At the end of the auction, the winning bidder does not receive the item nor are they required to pay. Such auctions are designed to emulate real auctions without concern over payment settlement.

Real auctions, on the other hand, involve auctioning real, tangible items. Bidders are also required to use their own money. When the auction finishes, the winner receives the item bid upon and must also pay the auctioneer the winning amount. Such auctions are used to gauge actual auctioning behavior, which can differ from simulated behavior as there are real risks and rewards.

Enforcing payment in real auctions is a complicated task. This often involves setting up accounts, providing insurance, gathering user feedback on sellers, etc. We choose not to tackle this topic in this article. Instead we leave the architecture open to implement any payment mechanism for real auctions.

Simulated auctions provide the system with a great deal of control. Each user can be issued an initial balance. When a bidder wins an auction, the amount of the winning bid is deducted from the bidder's balance. A user is unable to submit a bid for an amount greater than their balance.

The user entity in the entity-relationship diagram in Figure 5 can easily be modified to include this behavior. Figure 8 illustrates the amended user entity. The red section depicts a new attribute called *balance*. The balance attribute contains the current balance of the user. When a bidder wins an auction, his/her balance attribute is updated.

Conclusions

Our auction model is specific to online English auctions and it is simple and concise compared to previous literature on auction design. Our system is modelled using an object-oriented approach and is used to test the efficiency and practicality of security and anonymity mechanisms in online auctions. RAS has been used to perform hundreds of simulated and real auctions.

RAS is being extended to other types of auctions. For example, we have implemented a form of auction referred to as a continuous double auction (CDA). This allows many buyers and sellers to continuously trade goods. The best known example of a CDA is a share market. A CDA is inherently more sophisticated than an English auction. This involves modifications to the database schema and the winner determination procedure. We are in the process of documenting the software design considerations of RAS's CDA mechanism.

Acknowledgments

The authors would like to thank Dr. Sharith Trevathan for her contribution to this article.

References

1

Peter Coad and Edward Yourdon. "Object-Oriented Analysis", *Yourdon Press Computing Series*, Second Edition. 1991.

2

eBay <**http://www.ebay.com**>.

3

Matthew Franklin and Michael Reiter. "The Design and Implementation of a Secure Auction Service", *IEEE Transactions on Software Engineering*, vol. 22, Pages 302-312. May 1996.

4

M. Kumar and S. I. Feldman. "Internet Auctions", in *Proceedings of the 3rd USENIX Workshop on Electronic Commerce*, Pages 49-60, August 1998.

5

Abraham Silberschatz, Henry Korth and S. Sudarshan. "Database System Concepts", *McGraw Hill Computer Science Series*, 1996.

6

S. Stubblebine and P. Syverson. "Fair On-line Auctions Without Special Trusted Parties," in *Proceedings of Financial Cryptography 1999, (M. Franklin ed.)*, vol. 1648 of Lecture Notes in Computer Science, Pages 230-240, Springer-Verlag, 1999.

7

Jarrod Trevathan. "Security, Anonymity and Trust in Electronic Auctions" *ACM Crossroads Student Journal*, Spring Edition, Pages 3-9, vol. 11.3, 2005.

8

Jarrod Trevathan, Hossein Ghodosi and Wayne Read. "Design Issues for Electronic Auctions", *ICETE'05 - 2nd International Conference on E-Business and Telecommunication Networks*, Pages 340-347, October 2005.

9

Jarrod Trevathan, Hossein Ghodosi and Wayne Read. "An Anonymous and Secure Continuous Double Auction Scheme", *HICSS'06 - 39th Hawaii International Conference on System Sciences*, Page 125, January 2006.

10

uBid <**http://www.ubid.com**>.

11

Michael P. Wellman, Peter R. Wurman and William E. Walsh. "The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents", *Second International Conference on Autonomous Agents (AGENTS)*, Pages 301-308, May 1998.

12

Michael P. Wellman and Peter R. Wurman. "Real Time Issues for Internet Auctions", in *First IEEE workshop on Dependable and Real-Time E-Commerce systems (DARE 98)*, 1998.

13

Michael P. Wellman and Peter R. Wurman. "Parameterization of the Auction Design Space", Games and Economic Behavior, vol. 35, Pages 304-338, 2000.

14

Michael P. Wellman and Peter R. Wurman. "The 2001 Trading Agent Competition", *Electronic Markets*, vol. 13, no. 1, Pages 4-12, 2003.

Biography:

Jarrod Trevathan (**jarrod@cs.jcu.edu.au**) is a PhD student at the School of Mathematical and Physical Sciences, James Cook University. His research interests include security, programming, multimedia, and electronic commerce. Jarrod has also

taught Computer Science for six years.

Wayne Read (**wayne.read@jcu.edu.au**) is Head of School at the School of Mathematical and Physical Sciences, James Cook University. His research interests include modelling transport processes in porous media, series solutions to partial differential equations, and restricted occupancy models.