Design of a Real-Time Auction System

Hemantha Kumar P., Gautam Barua
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati 781039
email: g_barua@yahoo.com

Abstract

A number of auction systems are functioning on the Internet. Such systems however cannot handle the demands imposed by a high-volume, real-time auction system, with a large bidding population. This paper presents a study of a Tea Auction System. Based on this study, an architecture based on software agents acting on behalf of bidders, has been proposed, and a prototype implemented. This architecture can be used to design auction systems for other commodity markets such as coffee, tobacco, onions, rice, wheat etc.

I. Introduction

AUCTIONS over the Internet have gained popularity in the recent past. Compared to projections of various forms of electronic commerce, online auctions have been among the most successful medium. Online auctions are growing in popularity as a way of setting prices and distributing the reams of information required for complex transactions. These auctions are suitable for high-volume, low-fee business transactions.

The auction floor is a Colosseum, where the bidders try to obtain the best of the goods for a very reasonable price. These auction floors play a very important role in commerce and many industries, where transactions of goods takes place more frequently and in large quantities. Traditional auction floors insist that all traders, both the bidder and the auctioneer, be present geographically at the same place. The deployments of digital networks - LANs, MANs, and WANs, both wired and wireless, as well as other technologies that facilitate computer-to-computer communications help in accelerating the breadth and effectiveness of electronic commerce, and in particular auctions. This kind of environment is useful to auctions due to these reasons [3]

- these digital networks support inexpensive, wide-area, dynamic communications,
- tedious auction-mediated negotiations can be automated. Both the auctioneer, and the participating traders, may be represented by computational processes,
- these online auctions provide a mechanism for traders to obtain necessary goods at the same time from different auctions, which are geographically apart.

A. Auctions

An auction can be defined as the buying and selling of goods through a system of open public bidding where traders give bids according to a predefined scheme until only one valid bidder remains. Most auctions, like the *English* auction, provide a facility to share the goods among the bidders. The individual with the highest bid wins the item auctioned and earns the right to buy item for that price. An auction can be used to sell just about any item. Commonly, works of art, antiques, perishables (agricultural products), and even stocks are auctioned. There are a variety of auction systems that are currently being used all over the world. These auctions can be classified by whether they are single or double sided. Each of these types are further classified as sealed or public[4]. Dutch, English, Sealed-bid and Vickery auctions are some of the well know auction algorithms followed currently over the Internet. Each of these auction methods has subtle variations [11] such as:

- Anonymity, i.e., what information is revealed before, during and after the auction. For example, the identity of the bidders could be concealed. In a sealed bid auction the final winning prices could be kept confidential. In all auctions the amount of inventory may or may not be announced in advance.
- Rules for ending an auction. Open cry auctions may end at a posted closing time. Alternatively the auctions could be kept open so long as new bids continue to arrive within some time interval of the preceding bid. This interval would be several minutes in an Internet auction and a few seconds for an auction being conducted in a meeting room. One could also choose to close the auction if either of the above two conditions is met or only when both conditions are met. Dutch auctions could close at a pre-specified time, when all the inventory has been sold, when the price has fallen to a pre-specified level, or at some combination of these three conditions.
- Once the bidding phase is over, the bidders with the highest bids get the item being auctioned, but the price they pay could be the same as what they bid or lower. In a **Discriminative** Auction, also known as **Yankee** Auction, the winners pay what they bid. In a non discriminative auction people with winning bids pay the price paid by the winning bidder with lowest bid. Finally, in an auction for a single item, in a Vickrey Auction the winner pays the price bid by the second highest bidder. Vickrey auctions are also referred to as second price sealed bid auctions.
- Restrictions on bid amount. In all auctions the seller can specify the minimum starting bid. To speed up the bidding process minimum bid increments are often enforced. The bid increment is roughly proportional to the current bid, i.e.,

they are smaller for lower bids and larger at higher bids. The seller may also be allowed to specify a reserve price, which is a lower limit on price acceptable to seller. The buyers may know that a reserve price exists but they may not know what the reserve price is.

B. Software Agents

A software agent can be viewed as an autonomous software construction; i.e., one that is capable of executing without user intervention. An autonomous agent[11] is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future. Software agents provide assistance in performing (usually simultaneously) various tasks efficiently and promptly. These agents are ideally suited for use in network applications where the bandwidth is low.

Agents can be broadly categorized into two, *static* and *mobile* agents. Static agents reside on the user's PC, and provide expert advice or services. In the other case, the agent is incorporated with mobility which allows it to execute commands while living on a remote server, only reporting back to its home base when the given task is accomplished. Independent, agent learning, cooperation and reasoning capabilities are often associated with the notion of an intelligent agent[1]. Several other attributes can also be found in the literature with regard to agents [11]: cognitive, mobile, flexible, adaptive, and so on. Obviously, not all of the above properties need be actually present in an agent.

The Internet and the World-Wide Web, with their open, world-wide, and ease-to-use nature, have become an increasingly important channel for e-commence. The Internet is likely to spawn many new markets. Perhaps the most promising application area for software agent technology is e-commerce, where the agents can mimic the way people conduct transactions – business-to-business, business-to-consumer, and consumer-to-consumer. One that is particularly attractive for multi agent technologies is network-based trading. Intelligent Internet software agents are a new class of software agents that act on behalf of the trader to negotiate and automate complex tasks.

II. Motivation: The Tea Industry

The tea industry has been the motivation for this project. The tea industry is one of the biggest industries in India. Different varieties of teas are grown in different parts of the country. Tea grown in one region of the country is now being sold all over the country, through auctions. Buyer companies, based at different parts of the country send their *human agents* to these auction centres. These *human agents* try to obtain the best tea for the cheapest price possible. Trading in these auction centres follow the famous *English auction.* Tea is auctioned in lots. These lots are auctioned through a system of open public biddings where traders give successively increasing bids until either the auctioneer is satisfied with the price or only one bidder remains. The highest bidder is entitled to the good which is then purchased from the seller for the winning bid price. The time taken for a lot of tea to be sold after it is manufactured is very long, about a month. This is due to the huge amount of post auction work which includes money and goods transcations.

These old methods of business are no longer suitable for use in today's competitive environment. New tools and technologies are required to allow the industry to grow and thrive in this new era of deregulation. New methods for buying and selling tea, and arbitrage are required. Technologies for automating these methods are also required. Software agents over the Internet provide a promising mechanism for implementing this complex system.

A. The Auction System

Tea is auctioned into lots. Each lot consists of packages of a certain grade. The broker[1] puts the lots in one or more sales, Each broker prepares a sale catalogue, which is distributed to all the potential bidders along with the tea samples. Also, each broker produces a valuation sheet for the lots in the catalogue. In the sale catalogue each lot is given a lot number, which is unique with respect to a particular broker and for a particular sale. Auction for a sale is held at the auction centre for two days in a week. The auction can also extend to a third day if necessary. The other three and half days are utilized for accounting and completing the post auction transactions. The auction dates and the time schedule are released by the auction centre. The auction centre also specifies the prompt dates both for buyers and the sellers. All post auction transactions have to be finished before these prompt dates. Following are the rules that all the traders[2] have to abide by

- all the brokers have to strictly adhere to the time table. However, to properly determine the price for future auctions the first broker in the CTC, Leaf and Dust sections of the sale may if necessary avail an extra 10 minutes and the second broker an extra 5 minutes. Also, every broker would get an extra 3 minutes for every 100 lots auctioned.
- the auctioning rate should be not more than 1.3 min/lot.
- grace time would be provided for each broker. The selling broker should make way for the next broker once the alloted time, inclusive of grace time, is used up.
- a particular lot can be shared between two bidders, at the highest bid price, only if the lot contains more than 30 bags and between three bidders if the lot contains more than 40 lots.

[1]In this auction system the broker is the auctioneer himself.

[2]both the auctioneer and the bidders

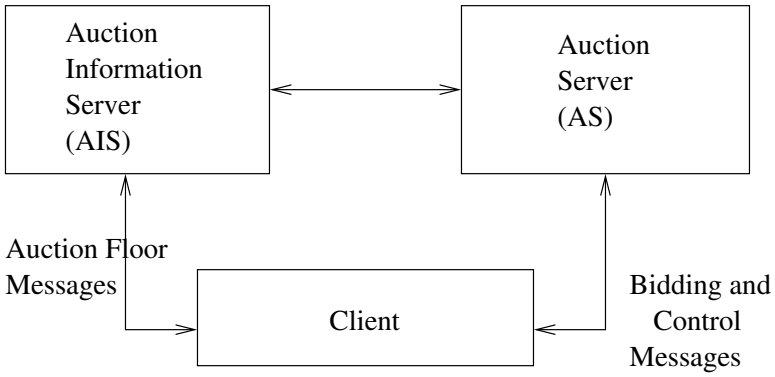

Fig. 1. Basic Architecture of the Auction Interface

III. Technical Challenges

Existing auctions are widely used for trading goods, whose price is less effected by the time taken to sell them. These goods are auctioned for a long period of time ranging from a few hours to few days. The existing auction systems are designed using existing technologies like *HyperText Transfer Protocol (HTTP), Java Applets, JavaScript, and Secure Socket Layer (SSL)*, that drive the Internet. These technologies follow the well known client-server architecture.

The client-server system over the Internet works well for auctions which are conducted over a relatively long period of time. Here the client is provided with an interface, implemented as a Java Applet in the browser, for the auction floor. The client carries out the bidding process through this interface. The server recieves the bidding information which is usually the bid value, and makes decisions accordingly. Later the server broadcasts the bidding information and the decision to all the clients that are involved in the auction. For realising real-time auctions where bidding and decision making is done within fixed amounts of time, this auctioning architecture requires many changes to meet the demand for a highly reliable, high bandwidth connections. Important issues that are considered while desinging such an architecture are given below.

- The network speed does not remain constant throughout the auction period, due to the network traffic. Such variations in network latencies can be surrmounted by conducting the bidding process locally.
- Internet connections are still unreliable. Network link failures during bidding process can be avoided by automating the bidding process on the auction server.
- The tea auction is a real-time system in which one lot is auctioned at a rate of 1.3 min/lot. The number of active bidders which is about 70, produces a large volume of auction data. Sending every bid to all the active bidders, in the absence of multicasting support, is not going to be possible within the given time constraints.

In order to overcome the problems of time, bandwidth and reliability, an auction system driven by software agents has been designed. This auction system realizes the real-time auction carried out at the Guwahati Tea Auction Centre. A similar architecture can be used for other real-time, commodity auctions.

IV. Agents and Agency

While agents are useful and powerful computational entities and a single agent can be used to perform a wide variety of tasks, agents are most useful when multiple agents communicate, cooperate and collaborate to solve complex problems. A collection of agents that work together to provide some service is called an *agency*[5].

Each agent in an agency has some specialized task that it performs. In the agency defined in the following sections, we define the bidder and the auctioneer agents that are delegated responsibilities by the various traders involved in trading tea. These agents communicate and cooperate with each other to implement an electronic auction house. In this auction house, the auctioneer agent auctions the tea to the bidder agents. The focus of this project was on developing an agency that would emulate the real life tea auction process.

The agency described in the following sections depicts a simplified version of the real life auction centre. The agency demonstrates the efficacy of the agent-based auctioning. The agency consists of one auctioneer agent representing the auctioneer and a number of bidder agents representing the clients.

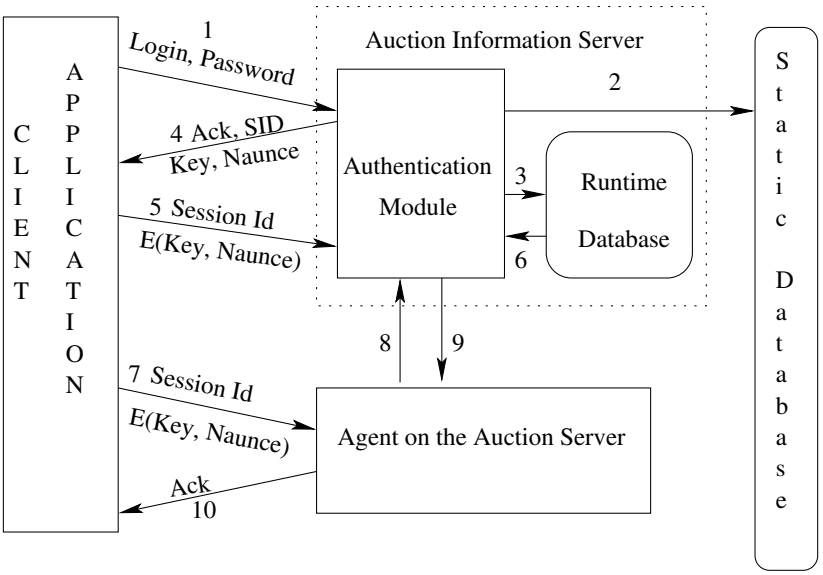

Fig. 2. The Authentication Protocol for the Auction System

A. The Architecture

Software Agents play an important role in this auction architecture. Since the tea auction system is a real time system, delay due to the network traffic is not acceptable. This delay can be surmounted by having software agents executing on the server to carry out the bidding, on the server, on behalf of the remote client. The agent receives the necessary bidding data for all the lots from the client, before the auction is started. Since, the agents are executing on the server, there is no delay in the bidding process due to the network traffic. Agents at the server side request for the data at periodic intervals of time, thus reducing the network traffic. The system is divided into two distinct interfaces, the *web interface* for obtaining pre auction data like the sale dates and catalogues. The *auction interface* provides access for the client to control the agent on the server.

The *auction interface* accesses and modifies a *run-time database*. This database maintains the current state of the auction. The state of the auction is defined by the following parameters

- the active lot number that is being auctioned.
- the time elapsed since the beginning of the auction of the current lot.
- the bidding messages.
- the current active agent and its bid value (if any).
- the highest bid value and the agent who made the bid.

The basic design of the *auction interface* is shown in Figure 1. This interface has been further divided into two servers, the Auction Information Server (AIS) and the Auction Server (AS). In brief, the AIS provides information regarding the current status of the auction floor, while the AS provides the resources for the auction. This division of work helps in effective utilization of time for the actual auction process.

B. Authenticating to the Agency

The client has to authenticate itself before triggering the agent or entering into the agency. This process of authentication should be carried in a secure environment. The proposed authentication protocol is shown in Figure 2.

The authentication is done in three phases. In the first and second phases the client authenticates to the AIS. In the first phase the client authenticates itself to the AIS over an SSL connection [8]. The second and third phases of authentication is completed over the standard *Transmission Control Protocol (TCP)*. These two phases are completed in 8 steps as shown in Figure 2.

The first phase of authentication is compulsory for all the clients. The client initially makes a connection to the AIS on a pre-declared port, using a *SSL Socket* instead of the standard Berkeley sockets, for Unix like environments or WinSocks for MS Windows. The following four steps complete the authentication process of the client with the AIS.

- After the connection has been established, the client transmits the login and password to the AIS. This connection is a SSL connection.

• The *Authentication Module* of the AIS refers to the static database and checks the validity of the data obtained.
• If the data obtained proves the identity of the client, the *Authentication Module* creates an entry for the client in the runtime database.
• Then it transmits the *Session Id (SID),* a randomly generated unique number *Naunce (N),* and the secret which the client and AIS share for the current session, *Key (K).*

These four steps thus, complete the first phase of authentication to the auction server. After obtaining the acknowledgement from the AIS the client closes the SSL connection. The SID has a certain life time, which is configurable. This first phase of authentication creates an identity for the client on the server for a particular amount of time. the client can use this identity for authenticating itself to the AIS and AS separately.

The second phase is an optional phase. In the second phase, the client connects to the AIS again, but to a different port. This is a normal TCP connection. Now the client transmits the *SID* in plain text and *(N-n)* encrypted with K using a lightweight encryption algorithm. The n is the number of times the client has connected to either the AIS or the AS with the current SID. If the client has connected to the AIS twice and to the AS once then n takes 3.

After this second phase of authentication the client can view only the current status of the auction floor which includes all the messages on auction floor. However, no bidding is done on behalf of the client because the agent for the client has not yet been created. The client has to separately connect to the AS to trigger a bidding-agent which is done in the third phase of authentication. The following steps complete the third phase of authentication.
• The client, after successfully authenticating to the AIS, can connect to the AS on another pre-declared port. This time the connection is made similar to the the connection done in the second phase while connecting to the AIS.
• This data is passed by the AS to the *Authentication Module* in the AIS. The *Authentication Module* checks the validity of the client from the data available from the runtime database. If the client has properly authenticated the value of N is reduced by one as it is done in the second phase.
• The *Authentication Module* returns the acknowledgement to the AS.
• If the acknowledgment is positive the AS creates an Agent for the client and sends the same message to the client. However, if the acknowledgement is negative no Agent is created and only the message is transmitted to the client.
After this phase of authentication the client can send the bidding data to the agent created. All the bidding data is encrypted using the same secret key K. The newly created agent is delegated the job of bidding on the auction floor on behalf of the client.

C. The Auction Information Server (AIS)

Apart from authenticating the client another important task of the AIS is to provide all the necessary information regarding the auction to the remote client. In the real life auction, nearly 70 to 100 bidders actively participate in the bidding process. Also, in a day, 500 to 600 lots of tea are auctioned. This generates large volumes of bidding messages. Transmitting all these messages to the clients who are currently logged on to the AIS would certainly slow down the system. In order to alleviate this problem, messaging or AIS agents are assigned to each client logged on to the AIS. Figure 3, shows the high level architectural components of the AIS.

When the AIS and the AS are initialized, the AS connects to the AIS internally. An identification number is assigned to each message obtained from the AS, before inserting it in the message queue. Each message in the queue also contains a counter. When the message is inserted in the queue the counter is set to zero. As the AIS agents read the messages the counter is incremented and the last AIS agent that reads the message also removes it from the message queue.

The AIS agent transfers the auction floor messages to the clients depending upon the activity of the client. This means that if the client has actively participated by bidding for the lot, all the bidding messages pending till then are transmitted to the client as a single message. Otherwise, the message is not read from the message queue. If there are three or more than three pending messages all of them are transmitted to the client as a single message. This reduces the number of messages and thus reduce the network traffic. The value three can be changed. The last agent to read the message removes the message from the message queue. Each message is signed with the current value of the *Naunce* for that session.

Another task of these messaging agents in the AIS is to transfer messages among the clients. This is quite similar to the normal chatting. By default, when the client authenticates to the AIS the agent associated to the client becomes the member of the auction floor room. This means that any client can send a message to any other client currently logged in the auction room. This is helps in negotiating the price of some particular lots.

D. The Auction Server (AS)

The AS is the place where the actual auctioning of the tea takes place. The Block level architecture of the AS is shown in Figure 4. The AS can be viewed as a collection of agents. While there is only one auctioneer agent, the rest of the agents are the bidder agents making bids on behalf of their clients. These agents are created when the client successfully authenticates to the AS.

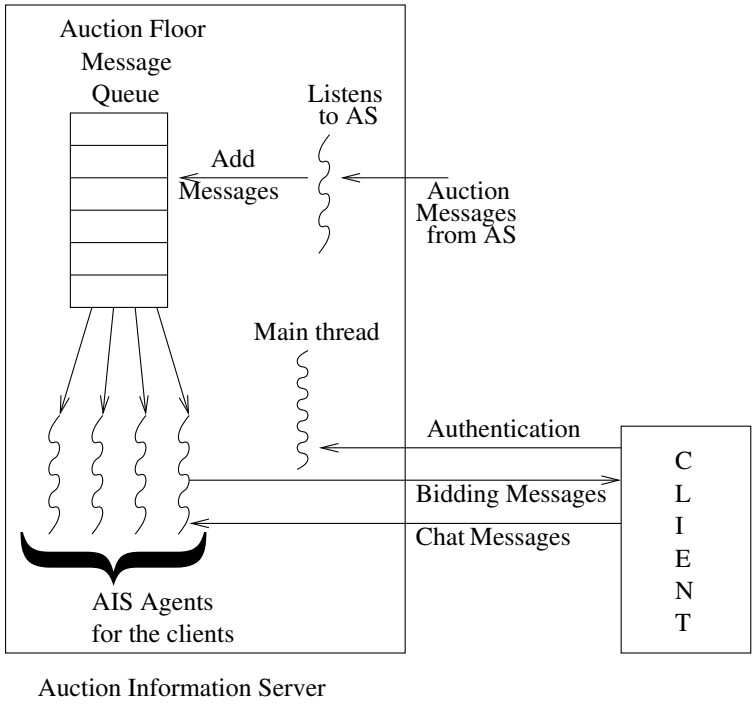

Fig. 3. The Auction Information Server (AIS)

The agents are semi-autonomous. The clients inform the agents the lots for which bids have to be made. The agents are also provided with some parameters for each lot to enable them to make valid bids.

As discussed in IV-B, each message from the client to the agent and vice versa contains a message id. This prevents replay attacks on the agent.

E. Client and Agent Communication in AS

All the messages transmitted by the client can be classified into four major types.

• Bidding Data Message: This message contains the bidding data of the lots to which the agent has to make the bidding for. The bidding data for each lot comprises of

– Minimum/Base Bid: It is the minimum value the client has estimated for the lot. If the agent is the first to bid for a particular lot it makes a bid for this amount.

– Maximum Bid: This is the maximum price the client can afford to buy the lot for. The agent can will never make a bid which is higher than this value.

– Tolerance: If the value of *bid* in Equation below is non zero then the agent makes a bid for the lot.

$$bid = \begin{cases} min\,(Maximum\,Bid,\, price\,quote + tolerance) \\ Minimum\,Bid,\, if\,the\,agent\,is\,the\,first\,to\,bid \\ 0,\, otherwise \end{cases}$$

– Alarm: If the value of *alarm* in Equation below is non zero the agent sends an alarm message to client.

$$alarm = \begin{cases} price\,quote,\, if\,bid \geq Alarm \\ 0,\, otherwise \end{cases}$$

– Request Share: If the agent has reached the maximum bid value it can issue a request for sharing message, if this value is non zero. This message is issued to the auction floor. This value gives the number of bags, in the current lot, requested for sharing.

– Accept Share: If the client is interested in sharing a particular lot this value is set to the minimum number of bags in the lot, the client can share. A zero value implies that the client is disinterested in sharing the lot.

These are the parameters used by the bidding agents. The auctioneer agent requires a different set of parameters as data. These include

– Minimum price: It is the minimum price for a lot. Any bid lesser than this value is regarded as an invalid bid.

– Next Minimum price: If no bidder agent is interested in the lot because the *minimum price* is high for the bidders, the auctioneer agent can reduce the minimum value to this value. All the agents are notified after the change is made.

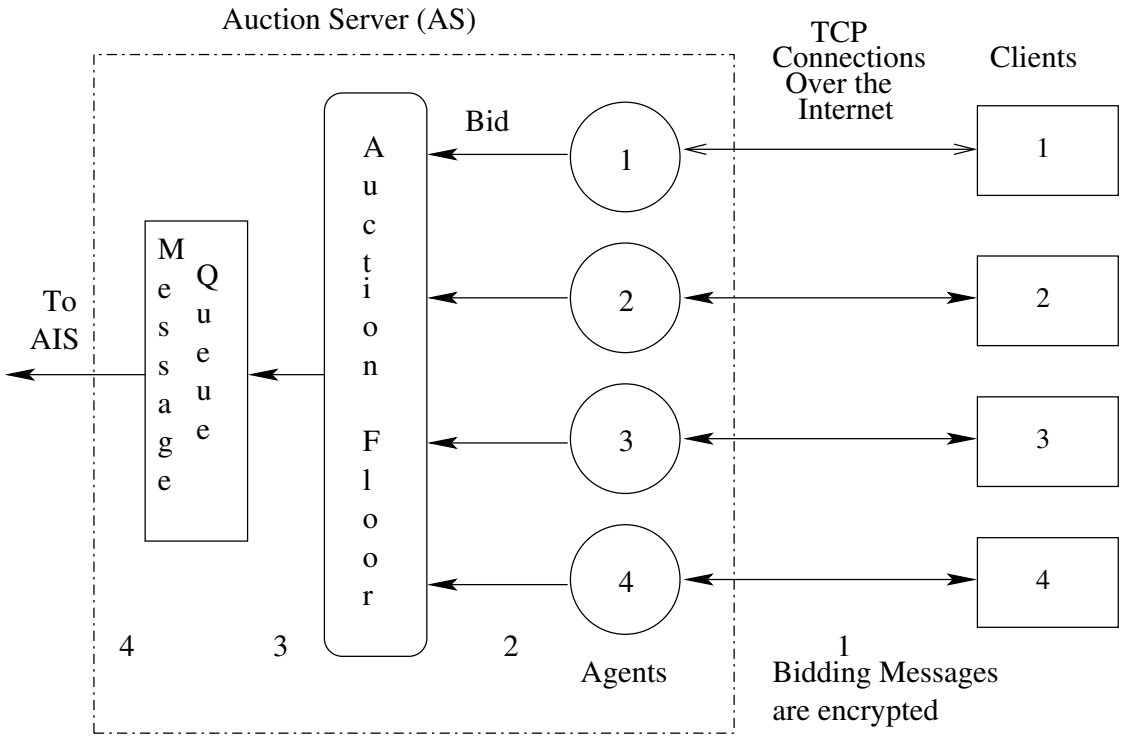

Fig. 4. Basic Architecture of the Auction Server (AS)

• Control Message: These messages include the replies for the alarms the agent has sent to the client in the past. These messages inform the agent whether to increase its maximum bid value or issues a request for sharing message.
• Agent Settings Message: Before the agent can request for any data from the client it has to be configured. This configuration includes:
– Number of Lots: The total number of lots the agent can request each time from the client.
– Alarm Lots: When the agent is left with these many number of lots it would request the client for the next set of data.
• Data Change Message: This message is generated when the client needs to change the data that has already been transferred to the agent on the remote AS. The normal *Bidding Data Message* is encapsulated into this message.

While the *Bidding Data Message* and the *Control Message* are generated due to some previous request of the agent, the *Agent Settings Message* and *Data Change Message* are triggered as a result of changes in the data made by the client. The agent, on the other hand, has three different messages for the client.
• Current Status: This message is generated every time the client is connected to the agent. This message contains the current status of the Auction Floor. The broker (auctioneer) conducting the auction, the tea type and the current lot number are the fields of this message.
• Next Set: When the agent has remained with data for only *Alarm Lots* number of lots it generates this message. This message contains the next lot number
• Alarm: When the agent has encountered the alarm value while bidding it will generate this message. This message contains the *price quote* for the current lot.

V. Implementation

The auction system can be viewed as a collection of well defined objects interacting with each other by exchanging messages in a fixed format. The most important objects are the AIS, the AS, messaging (AIS) agents, and bidding/auctioneer (AS) agents. Also the messages are structured. These requirements demands for a powerful object oriented programming language like C++ and Java for realising these objects. In the auction system messages are passed concurrently between the agents and the auction floor, among agents through the auction floor, and the agent and its client. This also makes it necessary that the programming language should support multithreading and should have advanced network programming facilities. Taking into consideration all the above mentioned points the entire auction system (AIS, AS and the internal agents), and the client interface is implemented in Java.

The auction algorithm is implemented in two different ways. In the first method, each agent in the AS is implemented

as a Java thread. In the second method, these agents are implemented as normal objects.

A. Method I

In a real life auction, all the bidders come to know the *price quote* at the same time. They react to the previous bid within few seconds. To emulate the same situation, the auction floor is declared as an *auctionObservable* object and each agent is declared as an *auctionObserver* object. These are similar to the standard Java *Observer* and *Observable* objects [13]. The agent threads are added to the list of observers on the auction floor at the time of their creation. When an event occurs on the auction floor, all the threads in the list of observers are notified of the change. This notification makes all the blocked agent threads runnable, which execute the *update* procedure. This *update* procedure contains the code for bidding. The Java runtime executes these runnable threads in random manner.

This implementation of the auction algorithm approximately automates the real life auction system. This implementation inherently possess randomness in choosing the agent thread for current execution, since the Java runtime scheduler randomly selects a runnable thread for execution. However the overhead involved while notifying, blocking, and waking of threads, is quite costly taking into consideration the number of agents present in the AS. Each agent has an extra thread blocked for I/O over the socket stream, which is also managed by the Java runtime scheduler.

At any instant of time, only one agent can make a bid, which in turn may change the state of the auction floor. Hence all the operations on the auction floor should be atomic. To obtain atomicity, the auction floor object is locked when an agent tries to change its state. This involves an extra overhead of obtaining and releasing locks. Also, any other thread trying to access the auction floor is blocked, since the entire auction floor object is locked[3].

B. Method II

The implementation discussed in the previous subsection was found inefficient. The reason attributed for this inefficiency was the overhead involved while preempting an agent thread by the Java runtime scheduler. Since time is critical in this auction, any delay occurred while executing the agents is not acceptable. To overcome this delay in execution of the agent threads, each agent is implemented as an object. These agent objects are created and initiated during the client authentication. A new agent is added to a vector of agents.

When the auctioneer agent starts the auction, the auction floor forks a *controlling* thread which executes the agents. This *controlling* thread picks the agents one by one in a random order and executes the *update* procedure of that agent. The priority of this thread is same as that of all the other threads. Once, the *controlling thread* completes its execution of the *update* procedure it places back the agent in the vector of agents. This continues till the auctioneer agent is interrupted by the timer which it has started at the beginning of the auction. This implementation has executed remarkably faster compared to the implementation discussed in the previous sub section. This is because preempting of threads occurs at quiet less number of times as there are no agent threads. The human decision and reaction time is very slow. Considering the time taken for reacting for an alarm message and the current Internet latency, which is in order of milliseconds, the time saved in the execution of the agents is quite valuable. Equal chances for bidding is ensured by picking the agents in a random order.

C. An Alternative

A further improvement in the implementation can be obtained by conducting the auction for all the lots partially and notify the clients the status of the auction floor. The clients provide the entire auction data to the agents just before the auction starts. The agents are triggered when the auction starts and bidding for the product starts. At the end of the auction if any agent has set an alarm flag the auction is suspended by the auctioneer agent, and the auction for the next lot is started. In this manner, the auction for a fixed number of lots is completed. After the auctioning by the agents is completed, the clients are now informed of the events that happened on the auction floor and the bidding for the suspended auctions is done by the clients themselves. Messages are exchanged between the agent and the client only after the auction for the last lot is completed. This auction algorithm reduces the network traffic considerably. However, this is not implemented.

VI. Conclusion

The auction system at the Guwahati Tea Auction Centre has been studied at length. The auction system has been parameterized formally and an electronic form of the system is proposed and partially implemented. The design, discussed in the previous chapters, proves that real time applications over the Internet can be efficiently implemented using software agents. Hence, the system is designed using software agents.

Bidders may initially feel insecure of the system and may set alarms very early. This will hinder the speed of the auction process. In order to get good results the bidders should have the alarms set near to the maximum price. This

[3]In Java locks are obtained over entire objects. Any thread accessing a locked object is blocked over the lock until the lock is released.

results in less alarms and saves more time which can be utilized for auctioning other lots. Experiments with the system are required to check if users adjust their behaviour once confidence over the system is established.

Implementing real-time auctions with a high volume of trading over a computer network is a challenging task. This paper has proposed an architecture using software agents to help meet the challenges. The Tea Auction System at the Guwahati Auction Center has been used as the application to build a prototype system.

References

[1] Ravi Kalakota, and Andrew D. Whinston, *Frontiers of Electronic Commerce*, Addison Wesley, First Edition, 1999.

[2] George Coulouris, Jean Dollimore, and Tim Kindberg, *Distributed Systems - Concepts and Design,* Addison Wesley, Second Edition, 1994.

[3] Michel 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).

[4] Peter R. Wurman, Michel P. Wellman, and Willian E.Walsh, *The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents,* In *Proceedings of the Second International Conference on Autonomous Agents,* p-301, May 1998.

[5] *Using Intelligent Agents to Implement as Electronic Auction for Buying and Selling Electric Power,* Alternate Energy Systems Consulting Inc., and Recticular Systems Inc.

[6] *The Fish Market Project,*
http://www.iia.csic.es/Projects/fishmarket/foundations.htm

[7] Carl M. Ellison, *Establishing Identity without Certification Authorities,* The Sixth Usenix Security Symposium Proceedings, 1996.

[8] Alan O. Freier, Philip Karlton and Paul C. Kocher, *The SSL Protocol, Version 3.0*, Transport Layer Security Working group, Internet Draft, November 18, 1996.

[9] *Software Agents in the Internet / Draft,* Olli Rysä,
http://www.tcm.hut.fi/Opinnot/Tik-110.501/1995/steganography.html

[10] Ross A. Gagliano and Martin D. Fraser, *Software Agents and the Role of Market Protocols,* Proceedings of the 35th Annual ACM Southeast Conference, p88-p90, April 1997

[11] Stan Franklin, Art Graesser. *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents.* Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996.

[12] R. Rivest, *The MD5 Message-Digest Algorithm,* RFC 1321, April 1992.

[13] Ken Arnold and James Gosling, *The Java^{TM} Programming Language Second Edition (The Java Series),* Addison Wesley, Second Edition, 1998.