Shared-Storage Auction Ensures Data Availability

Most current e-auction systems are based on the client–server architecture. Such centralized systems provide a single point of failure and control. In contrast, peer-to-peer systems permit distributed control and minimize individual node and link failures' impact on the system. The shared-storage-based auction model described in this article decentralizes services among peers to share the required processing load and aggregates peers' resources for common use. The model is based on the principles of local computation at each peer, direct inter-peer communication, and a shared storage space.

Hady W. Lauw, S.C. Hui, and Edmund Lai
Nanyang Technological University

Traditional auctions occur in an open-cry, first-price manner in auction houses, where participants outbid each other openly in real time; eventually, the bidder who submits the highest price wins the auction and pays that price in exchange for the item on auction. The Internet now facilitates electronic auctions, breaking the geographical barrier and time constraints imposed by traditional auctions. Current e-auction systems, such as eBay (www.ebay.com), Amazon.com Auctions (http://auctions.amazon.com), and Yahoo! Auctions (http://auctions.yahoo.com), are based on client–server architectures. They implement aggregated storage, computation, and network traffic at the server, which plays a role like an auction house in offering essential services for a fee.

Clients' reliance on the server has two main risks. First, information centralization makes the server an attractive target for hackers, thus exposing the whole system to the possible risk of abuse. The logical extension of this danger is that a downed server can cause the system to fail (client–server setups are particularly susceptible to denial-of-service attacks). On the other hand, peer-to-peer (P2P) systems adopt a network-based computing style that neither excludes nor inherently depends on centralized control points.[1] The machines that make up such systems can communicate directly between themselves without using central servers[2] (P2P nodes have equal capacity for sharing information). In systems such as Napster,[3] Gnutella,[4] and Freenet,[5] each user can take the role of producer and consumer of resources (files, processor cycles, hard-disk space, and so on).

Unlike retail, which is largely business-to-consumer, an auction is fundamentally a consumer-to-consumer or business-to-business activity. Like peers in a P2P network, auction participants have similar or equal standings[6] — a

 Published by the IEEE Computer Society 1089-7801/04/$20.00

participant could be a seller in one auction and a buyer in another. The equal participation in auctions and on P2P networks suggests that P2P could be a natural way to implement e-auctioning.

In this article, we describe a proposed P2P model for e-auctions based on shared storage – the shared-storage-based auction. Our main motivation for investigating such a model comes from P2P's potential to remove the server's centralized control and bottleneck effects. Our model goes one step further than existing P2P solutions by adopting a shared-storage feature that ensures continuous peer-data availability. Data is kept in a storage system formed by the aggregate storage resources contributed by peers on the network and whose utility is shared among the peers themselves. We have implemented a prototype system on the JXTA platform and evaluated its performance.

Shared-storage-based auction model

P2P storage systems have three types of memory components: directory, cache, and data store.[9] The directory maps a file identifier to one or more locations currently holding each file, which improves searching performance because the locations listed on the directory point to peers that are likely to return responses. The cache is used to temporarily store data items and increase their availability. The data store permanently maintains data items to ensure that at least one copy of a given item remains in the system.

Based on the memory components deployed, there are six possible architectures:

- data-store only,
- cache only,
- cache and data store,
- directory and data store,
- directory and cache, or
- all three.

Systems that use only the data store, such as Gnutella, must query many peers to search for data. Storage systems aided only by a cache, such as that used by JXTA Discovery Service to store and distribute advertisement files, reduce the number of peers to query. Freenet, which uses directory and cache, involves even fewer peers in queries, and Napster, which has a central directory, is especially efficient at searching (although it's also more vulnerable to attacks targeting the central directory).

A possible method of implementing a shared-storage-based auction model is to build auction capabilities into an existing system with a shared storage feature. The model's search efficiency would then depend directly on the shared-storage system on which it was implemented.

Architecture

Our model's main architectural components are peers that act as a collective shared-storage system. A seller uses the shared storage to publicize his or her auction, a buyer discovers auctions by searching among the storage entries, and a bid goes directly to the seller for local processing. In contrast to the client–server model, network peers rather than the server provide storage. Furthermore, because all communication is addressed directly to the intended recipient, we eliminate the "middleman" server, whose role would have been only to facilitate — but not participate in — the communication between peers.

Our architecture's main characteristics are shared network storage, local computation, and direct communication. Law-governed auctions also use the latter two features. (See the "Related Work in P2P E-Auction Systems" sidebar, next page, for more on law-governed auctions.) The principal difference is the means of information exchange: the law-governed auction uses an external registry, whereas with our method it is an intrinsic part of the auction system, made up of resources contributed by the peers themselves. In the latter, the auction system's utility is not reliant on external component availability.

To function as a repository for auction information, a shared storage system must have

- substantial capacity to store information on numerous auctions,
- continuous availability to enable round-the-clock retrieval,
- searchability to provide multiple matching responses to a query, and
- user mobility to enable retrieval from different locations on the network, letting users connect from different computers and still access the same data.

To ensure acceptable performance, the system should also support the cache memory component, at least. We could gain higher performance by using the directory in addition to the cache. A data store could be helpful by ensuring that at least one copy of a data item is permanently available on the net-

Related Work in P2P E-Auction Systems

To solve the client-server architecture's single point of failure and control problem in supporting e-auctions, several researchers have proposed peer-to-peer alternatives for e-auction models. In contrast to the shared-storage-based model, these alternatives either still maintain some centralized components or do not allow user mobility. Some examples include Enchère,[1] a law-governed auction model,[2] and Lightshare (www.lightshare.com).

Users access Enchère, a serverless distributed auction system, through autonomous workstations loosely connected as a network. The simple, totally decentralized design uses direct communication, as users communicate via network messages. However, it doesn't support user mobility — bidders must sit through the auction process at the same computer on which they sign on.

Fontoura and his colleagues developed the law-governed auction model to counter the fact that servers make the auction-process decisions in client–server environments.[2] To return the decision-making to the participants, an information-holding auction registry replaces the server. When registering an auction, the seller specifies how it is to be conducted. Buyers can query the registry to find out an auction's details and start bidding if they accept the rules laid down for that particular auction. However, a registry failure could disrupt the whole system.

Lightshare's main goal is to enable the sale of digital goods that require careful handling of copyright issues over a P2P network. The system uses a server, which acts as the single point of entry, but transfers a significant load to its clients. Users must create auctions through the server, but the seller's own computer saves the data after that. To search for an auction, buyers query the server, which in turn searches the sellers' storage space in real time. However, this design doesn't solve the single-point-of-failure problem because it still requires a server.

These systems use two different approaches to data storage. The local-storage approach can't support user mobility because the peer-specific data might not be accessible when a user moves to another computer. The central-storage approach is susceptible to single points of failure. Our shared-storage-based auction model attempts to solve these problems by storing redundant copies of peer-specific data on the distributed storage contributed and shared by peers on the network.

References

1. J. Banâtre et al., "The Design and Building of Enchère, a Distributed Electronic Marketing System," *Comm. ACM*, vol. 49, no. 1, 1986, pp. 19–29.
2. M. Fontoura, M. Ionescu, and N. Minsky, "Law-Governed Peer-to-Peer Auctions," *Proc. 11th Int'l Conf. World Wide Web*, ACM Press, 2002, pp. 109–116.

work, but it isn't essential because most auction information doesn't require permanent storage.

A problem with peer-contributed data storage is that peers can go offline without warning. Continuous availability can be assured only through some data redundancy and duplication. User mobility can be supported because data are accessible from anywhere on the network. For instance, users can move from computer to computer and still access their data as long as the computer being used is connected to the network.

Actors

All auctions include two types of participants: sellers and buyers. Software actors interface with these human participants and encapsulate role-specific task executions.

In our e-auction model, an auctioneer module encapsulates the auctioning tasks and selects execution-task details from the human seller. In turn, buyers use bidder modules to help locate auctions and submit bids. Bidder modules can contain intelligent bidding strategies that act as proxies to submit bids, without the need for the human buyer's constant intervention.

In the client–server model, modules equivalent to our software actors are instantiated only at the server. In our system, software actors operate at each peer, which means peers take over the computation load, distributing the load among themselves and not relying on a central server. Each peer can also act as an auctioneer and a bidder simultaneously in different auctions.

Services

We define a service as an interface with which software actors interact to facilitate auctioning and bidding activities. This encapsulation protects software actors from changes in task-implementation details. Our model includes three main services: *repository*, *bid*, and *presence*. Together, they let the shared-storage-based auction deliver equivalent properties to those in current client–server-based auctions.

Creating and publishing auctions. The auctioneer creates an auction by constructing a transaction record containing details, such as the starting bid, item description, auction end date, and so on. The auctioneer then publishes this record through the repository service, allowing potential bidders to find out about the auction through one of the search options provided by the repository service.

The repository service information exchange

assumes that the cache is supported by shared network storage; each peer maintains a cache to store its own and other peers' files. To publish information, a peer thus replicates its own files and stores them on many different peers. This increases the files' availability because they might be accessible from some peers even when others are offline.

Publishing also involves negotiating with other peers to keep the files in their caches. To keep the files up to date and prevent storage overload, each must have an expiration time, after which those peers still storing it will remove it from their caches. Occasionally, the peer publishing a file could refresh it to keep copies available in the network. To maintain consistency among the many copies of each piece of information, the publishing peer should specify short lifetimes and publish frequently so that up-to-date files regularly replace outdated ones.

Because the auctioneer is the sole authoritative author of an auction's transaction record, we must be able to verify that a published record is genuine. Digital signing ties the auctioneer's identity to each transaction record. In addition to preventing attackers from masquerading as auctioneers and publishing malicious content, it also disallows real auctioneers from repudiating their published records being used by bidders as a basis for bidding decisions.

Discovering an auction and its current state. Bidders find out about auctions by using the repository service to discover the most recent transaction records published by the auctioneers. Figure 1 illustrates a scenario in which a bidder finds information in other peers' caches. To locate information, the repository service first looks up the local cache. In the best case, a copy of that information can quickly be found on the local cache. The requestor searches other peers' remote caches if the local cache has no suitable response; expanding the search in such a manner also lets the requestor seek fresher or more varied responses. In the example in Figure 1, the Finder first contacts known peers in and outside the LAN to search in their caches. Peers 1 and 2 have no matching result and, thus, don't reply. Peer 3 finds a matching result and returns it to the Finder. Peer 3 might also forward the query to Peer 4 if the query has not been forwarded more than a certain number of times. If Peer 4 also had a matching result, it would respond directly to the Finder.

In addition to monitoring the auction's state (as published by the auctioneer), we must monitor other bidders' activities. Although it is the auctioneer's responsibility to stay connected for as long as possible to serve potential bidders, an auctioneer might need to go offline temporarily in some cases while bidders are serving bids. Though not yet confirmed by the auctioneer, these bids might represent important information — for example, the amount of the current highest bid. To address this, bidders queue their bids locally when the auctioneer is offline, while publishing their bidding intentions to other bidders through the repository service. Thus, each bidder can monitor other competing bids until the auctioneer is back online.

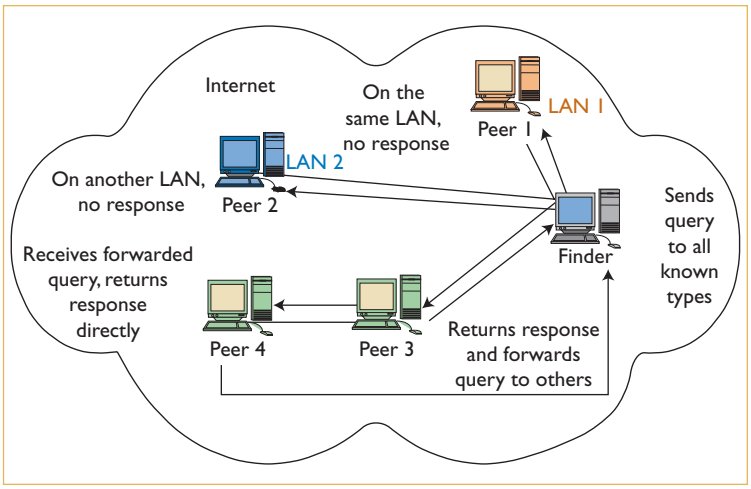

Figure 1. Finding information remotely. A peer, labeled Finder, initiates a query, which is forwarded to other peers up to a certain number of hops. Any peer that caches the information being queried responds directly to the Finder peer.

Sending a bid to the auctioneer. The bid service encapsulates the exact mechanism of how a bidder communicates a bid to the auctioneer. It formulates and delivers a bid message directly to the auctioneer, overcoming potential obstacles such as firewalls, which the service bypasses by using one or more intermediaries (peers on the same LAN but outside the firewall that could help forward the messages to the intended recipient).

To account for peers' transient presence — any peer could go offline at any time or come online from any network location — our model's presence service asserts each peer's identity, informing others of its current status and network location. Each e-auction human user has a unique identifier, such as an email address, which identifies this user to other users on files containing information on this user's presence and auctions. Using this unique identifier, the human user can still be identified even if he or she connects to the network from dif-

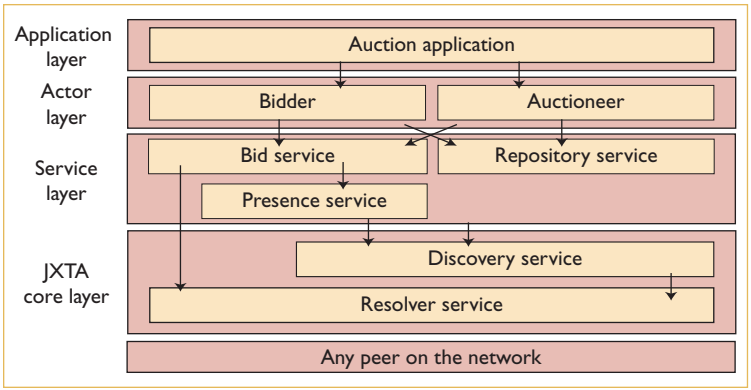

Figure 2. E-auction layered architecture. The system architecture consists of four layers, with the upper layers running more coherent and complicated tasks, such as buying and selling, by using more basic services provided by the lower layers, such as exchanging messages and finding information.

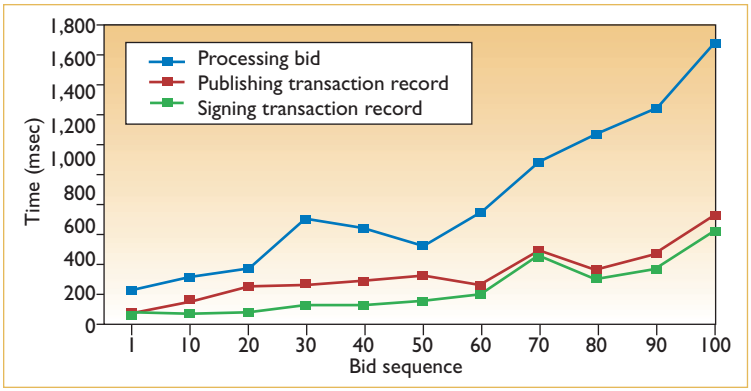

Figure 3. Effect of publishing and signing on bid processing. As an auction goes on, more bids are accepted and the bidding history gets longer, resulting in a larger transaction record. It gradually takes longer and longer to publish and digitally sign the growing transaction record, leading to longer response time. Here, we measured the time taken to process the first bid, the second bid, and so on, up until the 100th bid for the same auction.

ferent network locations. Consequently, because the user's network location could change, we must first dynamically map the user's unique identifier to his or her current physical network location. To perform address resolution and presence detection, the presence service running on each peer periodically publishes the following information about the user currently residing on that peer: unique identifier, current network location, and presence status (online or offline). This information also includes an expiry time to ensure freshness. Before submitting a bid, the bidder must first discover a fresh piece of the auctioneer's presence information to find out the auctioneer's latest known network location.

Winning the auction. As in real-world auctions, an e-auction bidder must submit the highest bid to win. Because bids go to the auctioneer and the other bidders, a bidder can correctly expect to win an auction after submitting the highest bid. We verify this by monitoring the auctioneer's final published transaction record, which declares all the bids made and who the final winning bidder is. To ensure nonrepudiation, bidders digitally sign each bid message they send to the auctioneer.

Performance Analysis

We implemented our system on the open-source JXTA platform, which is released as an open standard. We can readily use the cache-supported P2P storage system available in the form of JXTA's Discovery Service. Figure 2 shows our system's four-layer architecture: *JXTA core*, *service*, *actor*, and *application*. The core layer consists of core services provided by JXTA that use storage and communication functionalities on the JXTA network. The service layer implements common functionalities commonly required by software actors (the bidder and auctioneer) for publishing and finding auction-related information (repository service) or the user's current network location (presence service), as well as for exchanging bids (bid service). The actor layer defines two role-specific modules: Auctioneer to play the role of a seller and Bidder to play the role of a buyer. The application layer integrates all the services and layers into a single application and provides a user interface for human users. Because each self-contained layer interacts with its lower layers through a set of well-defined interfaces, it is not coupled to the particular implementation of the lower layers; it is therefore protected from having to change its implementation as a result of changes to its lower layers.

Our e-auction system is written in Java JDK1.4.0, running on top of the JXTA-J2SE stable release of 8 February 2002 (build 49b). As a result, it should run on any machine for which a Java virtual machine is available. To test the computational load of processing incoming bids, we ran a peer playing the auctioneer role on a PC (with Pentium III 1 GHz and 256MB SDRAM) running Windows XP, and measured the time it took to process bids coming from multiple competing bidder peers.

Processing Bids

Bid processing is the largest factor in system performance.[10] The system suffers some communica-

Table 1. Summary Application-Level Performance Comparison.

Performance criteria	Shared-storage-based	eBay	Enchère	Law-governed	Lightshare
System availability	Graceful degradation	All-or-nothing	Sessions of a few hours per day	All-or-nothing	All-or-nothing
Efficiency of search method	Acceptable efficiency	More efficient than shared-storage-based	Not applicable	More efficient than shared-storage-based	Less efficient than shared-storage-based
Quality of search results	Updated as currently known to information owner	Updated as registered by information owner	Not applicable	Updated as registered currently known to by information owner	Updated as information owner
Information control	Owner	Server	Owner	Registry operator	Owner

tion costs as bid submissions and responses move from bidder to auctioneer and back. Computation costs occur in packing and unpacking message information and in processing bid submissions and responses.

The auctioneer must compare each incoming bid against the current highest accepted bid value as well as against other competing incoming bid values and compute whether to accept it. The auctioneer then updates the auction's status by publishing a new transaction record. As Figure 3 illustrates, processing initial bids generally takes fewer resources than subsequent bids. The main activities that dominate resource usage during bid processing are signing and publishing new transaction records. As more bids arrive, the transaction record's size grows, resulting in more bytes to be digitally signed; this explains the proportional increase in the time required for signing. Similarly, publishing a transaction record involves I/O operations and network communications among peers to update multiple local and remote caches. This also consumes resources proportional to the transaction record's size.

If we assume that only a few auctions receive hundreds of bids and that only a few bidders generally compete in an auction's late stages, this trend doesn't pose a serious problem. If many bidders were to adopt a wait-and-see attitude, however, these assumptions might be invalid.

Application-Level Performance Comparison

We need qualitative aspects to analyze how the shared–storage-based model compares with other models. Table 1 shows a summary of the application-level performance comparisons for some existing e-auction systems. We compared our shared–storage-based system against eBay's client–server model and the Enchère, law-governed, and Lightshare P2P models, looking at availability, search speed, response quality, and information control.

The comparisons are qualitative as, at the point of writing, some of the other systems' implementations were not available for direct comparison. For example, Lightshare was still being developed; Enchère was a very new system, and its implementation included hardware prototypes we didn't have access to.

System availability. In a shared-storage-based model, peers can log on or off at any time. Sufficient redundancy means that an auction can be constantly publicized. The system's usability gracefully degrades as peers leave the system, up to a certain threshold, beyond which the system might no longer be usable. This could be due to the very long path lengths required to satisfy some queries or a lack of useful resources to attract peers to log on. Systems with a centralized component, such as eBay, law-governed, and Lightshare, rely on the central component's availability, which is either up with full functionality or completely down. As Enchère requires participants to sit through auction sessions, the system runs only during such sessions.

Searching method efficiency. P2P searching is generally less efficient than other approaches because information often is interspersed among many peers. However, our model's searching efficiency should be helped by the replication of data in multiple caches, increasing its availability, instead of storing it at only one location as Lightshare does. Searching over a centralized database, such as that done on eBay's database or a law-governed auction's registry, will be very efficient because all information is available locally and is likely to be indexed or sorted to optimize searching. Enchère does not include a searching functionality; instead,

distribution of information regarding auction sessions is performed outside the system.

Search result quality. We determine quality based on whether the search query's returned results are up to date and valid. Searching a centralized store of information (as in eBay or a law-governed auction) turns up responses that are as fresh as what the information owners have registered with the server or the registry. There could be a scenario in which a seller has new information that he or she has not yet registered; therefore, buyers cannot access this information. In general, a seller or buyer who is a direct auction participant would have the most updated information, such as possible changes in condition or the quantity of items. Therefore, searching in real time among peers' caches or data store, as in the shared–storage-based model and Lightshare, respectively, is likely to return the most updated information. Because Enchère does not have a searching functionality, this criterion does not apply.

Information control. A shared-storage-based model returns information control to each owner. This information decentralization prevents hackers from harvesting large quantities of information from a single location. It also lets individual information owners control how and to whom to reveal what kind of information. In contrast, users would not have any control over their information stored in a third party's centralized information storage, such as eBay's central server and the law-governed auction's registry.

Future Work

Our system's performance is encouraging, though not spectacular. Future work will include search optimization because decentralized data makes searching difficult. For example, we could use more developed searching methods for distributed environments such as the JXTA search service. JXTA search uses specialized peers acting as "hubs" that intelligently route queries to the most suitable information providers. These hubs can search more reliably than ordinary peers can, because ordinary peers have no way to determine which peers are better information providers. We also could adopt security practices via cryptographic techniques, but the system would incur significant computation costs. Other avenues we might explore include running auctions on mobile and interoperable computing devices such as portable PCs or PDAs.

References

1. L. Gong, "Peer-to-Peer Networks in Action," *IEEE Internet Computing*, vol. 6, no. 1, 2002, pp. 37–39.
2. D. Clark, "Face-to-Face with Peer-to-Peer Networking," *Computer,* vol. 34, no. 1, 2001, pp. 18–21.
3. C. Shirky, "Listening to Napster," *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, A. Oram, ed., O'Reilly and Assoc., 2001, pp. 21–37.
4. G. Kan, "Gnutella," *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, A. Oram, ed., O'Reilly and Assoc., 2001, pp. 94–122.
5. A. Langley, "Freenet," *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, A. Oram, ed., O'Reilly and Assoc., 2001, pp. 123–132.
6. M. Parameswaran, A. Susarla, and A.B. Whinston, "P2P Networking: An Information-Sharing Alternative," *Computer*, vol. 34, no. 7, 2001, pp. 31–38.
7. J. Banâtre et al., "The Design and Building of Enchère, a Distributed Electronic Marketing System," *Comm. ACM,* vol. 49, no. 1, 1986, pp. 19–29.
8. M. Fontoura, M. Ionescu, and N. Minsky, "Law-Governed Peer-to-Peer Auctions," *Proc. 11th Int'l Conf. World Wide Web*, ACM Press, 2002, pp. 109–116.
9. H. Hsiao and C. King, "Modeling and Evaluating Peer-to-Peer Storage Architectures," *Proc. Int'l Symp. Parallel and Distributed Processing*, IEEE Press, 2002, pp. 240–245.
10. T. Hong, "Performance," *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, A. Oram, ed., O'Reilly and Assoc., 2001, pp. 203–241.

Hady W. Lauw is a graduate student at the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include peer-to-peer computing and spatio-temporal data mining. He has a BEng in computer engineering from Nanyang Technological University. Contact him at hadylauw@pmail.ntu.edu.sg.

S.C. Hui is an associate professor in the School of Computer Engineering at Nanyang Technological University. His research interests include data mining, Internet technology, and multimedia systems. He has a BSc in mathematics and a PhD in computer science from the University of Sussex, UK. He is a member of the IEEE and the ACM. Contact him at asschui@ntu.edu.sg.

Edmund Lai is an associate professor at the School of Computer Engineering at Nanyang Technological University. His research interests include wireless ad hoc networks and digital signal processing. He received BE and PhD degrees in electrical engineering from the University of Western Australia. He is a senior member of the IEEE. Contact him at asmklai@ntu.edu.sg.