Peer-to-peer data preservation through storage auctions

Brian F. Cooper
College of Computing
Georgia Institute of Technology
cooperb@cc.gatech.edu

Hector Garcia-Molina
Department of Computer Science
Stanford University
hector@db.stanford.edu

Abstract

Digital archives protect important data collections from failures by making multiple copies at other archives, so that there are always several good copies of a collection. In a cooperative replication network, sites "trade" space, so that each site contributes storage resources to the system and uses storage resources at other sites. Here, we examine *bid trading*: a mechanism where sites conduct auctions to determine who to trade with. A local site wishing to make a copy of a collection announces how much remote space is needed, and accepts bids for how much of its own space the local site must "pay" to acquire that remote space. We define a spectrum of trading scenarios, ranging from a network of archives and digital libraries that trust each other, to a scenario where sites do as they please, including breaking the rules. Then, we focus on developing techniques for the scenarios where sites trust each other, although we discuss issues that may arise if sites are greedy or malicious. We examine the best policies for determining when to call auctions and how much to bid, as well as the effects of "maverick" sites that behave differently than other sites. Simulations of auction and trading sessions indicate that bid trading can allow sites to achieve higher reliability than the alternative: a system where sites trade equal amounts of space without bidding.

KEYWORDS: H.3 [Information storage and retrieval]: Systems and software - Distributed systems; E.5 [Files]: Backup/recovery; H.3.7 [Information storage and retrieval]: Digital libraries - systems issues

1 Introduction

Digital archives are sites charged with preserving important data over the long term. Making a few *local* backup copies of this information is not sufficient, since backup tapes break, compact discs decay and publishers go out of business (in addition to a host of other causes of data loss). Instead, archives need to replicate digital collections to other archives, so that there are always several good

This material is based upon work supported by the National Science Foundation under Award 9811992.

copies and a failure at one site does not mean that information is lost forever. Such a distributed archiving network can preserve important collections more effectively than simple backups over very long time periods (decades or centuries) [6, 14].

However, archives operate under two main constraints: the resources (such as storage space) they have are limited, and individual archives want to preserve their own autonomy and decision making. For example, a government agency may want to build a digital archive to preserve vital records. This agency may have a limited budget, and will not be willing to spend a lot of money buying and maintaining storage. Moreover, the agency is likely to be selective about the remote sites it will entrust with its collections, in order to protect private or sensitive information. Therefore, it is not possible to have a central decision maker allocating space in the most efficient way, since this reduces the autonomy of the local site.

We have developed a framework, called *data trading*, for replicating collections to achieve reliability, while allowing sites to decide where to replicate their collections and how many resources to contribute to the system. In data trading, two sites agree to "swap" collections, so that each site's data is replicated [8]. A series of such agreements between pairs of sites builds up a peer-to-peer trading network. Although each site is making local decisions for local benefit, the result is a global network dedicated to preservation.

In this paper, we focus on the negotiation of an agreement between sites. For example, site A may want to replicate a collection that is 100 GB large. Site A can contact site B and ask for a trade, and site B may respond that it is willing to trade if it receives 150 GB of site A's space in return. If site A contacts multiple sites asking for trades, then site A will receive multiple such "bids," and can pick the lowest or most attractive bid. Thus, an agreement may be concluded between site A and some other site C, where site C gives site A 100 GB, and in return site A gives site C 85 GB. This auctioning process gives sites the freedom to set their bids using any strategy that improves their ability to safeguard their data.

Our work draws upon concepts developed in related data replication systems. Figure 1 shows a schematic classification of data management schemes, including our work and some other sample

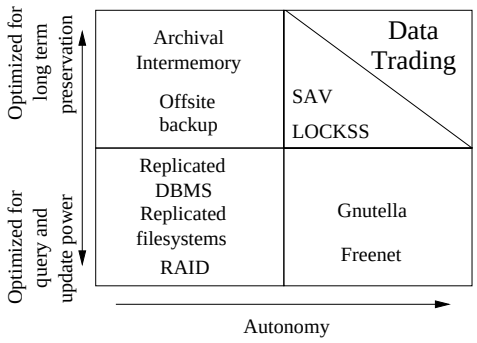

Figure 1: Classification of data management schemes.

systems. This classification divides schemes based on the amount of autonomy given to participating sites (horizontal axis) and whether the system is optimized for query and update performance, or for long term preservation (vertical axis). Our work is focused on the upper right box in the figure; that is, our main goal is to ensure reliability while preserving site autonomy. Such a *community-based replication system* necessarily makes different decisions than a system that can centralize control in one place, or that places data close to users in order to improve efficiency. Several systems, including SAV [6] and LOCKSS [27], can be classified as community-based replication systems. This paper discusses how such systems can allocate resources for the most reliable replication using auctions.

The concepts behind auctions, bidding and market oriented systems have been well studied by economists and computer scientists. In auction theory, our mechanism would be classified as a *first-price, sealed bid* auction [23]: each bidder submits a bid but does not know the other bids, and the winner pays the "first-price," which is the amount the winner bid. Ferguson et al note that in order to apply auction theory to a specific problem, several design questions must be addressed, including how to determine the value of resources to participants and how to conduct the auction as a distributed protocol [12]. These are some of the questions we address for the specific domain of reliable data replication in this paper.

Other distributed computing systems [13, 24, 31, 32] have used market-oriented principles (such as auctions) in order to allocate resources. Our work differs from these previous systems in several ways. First, most systems have a concept of "money" distinct from the resources that

are being bought and sold. In our system, there is no concept of "money," and resources are traded directly. This is because the location of the resource (e.g., at a remote site), rather than the resource itself, is the source of value. A barter system is simpler and more appropriate for an autonomous, peer-to-peer network than a system that requires some central entity to control the money supply. Moreover, given the very long time periods we are targeting (decades or centuries), it is unreasonable to expect that a central bank or currency will remain stable.

Second, many market-oriented systems assume a clear distinction between producers and consumers, such that producers have different incentives and follow different policies than consumers. In our peer-to-peer system, every site is both a producer and a consumer in every transaction, and thus must follow a policy that reflects this hybrid role.

Third, market-based data storage and management systems are usually designed to maximize a metric of access efficiency, or to tune the system for the read/update ratio of data items. "Profit" is made when a decision is made that increases the system efficiency. In our system, the economic incentive system must be structured to maximize reliability, rather than access performance. Related work is discussed further in Section 7.

In this paper, we examine how bid trading works, and evaluate policies that sites can use to construct bids. Specifically, we make the following contributions:

- We describe a mechanism by which archive sites can participate in auctions for the purpose of replicating their collections. This scheme is called *bid trading*.
- We examine different policies that sites can use for deciding when to call auctions, and how to bid when an auction is called.
- We present simulation results that show sites can increase the number of copies they make of their collections (thus improving their reliability) through bid trading. We also present results that show which policies are best under bid trading.
- We examine the effects of increased freedom on the reliability of the system.

Bid trading is intended to be used as the resource negotiation and allocation component of a general

distributed archival preservation system, such as that described in [6]. We focus on scenarios where sites know and trust each other, and do not intentionally act to game or subvert the system. For example, a group of archives, university libraries, and government agencies may form such a collective. We also discuss some issues that may arise if that level of trust is not present.

This paper is organized as follows. In Section 2, we describe the bidding process, including our model and the auction and bidding algorithms. Next, in Sections 3-5 we discuss policies for calling auctions and bidding, and ways in which maverick sites can deviate from "normal behaviors." Section 6 presents the results of simulation experiments where we study the various policies and maverick behaviors. In Section 7 we examine related work, and in Section 8 we present our conclusions.

2 Bid trading

An *archive site* is an autonomous provider of an archival storage service. The archive site takes responsibility for replicating digital collections deposited at the site by clients. A collection is a set of related digital material, such as issues of a digital journal, scientific measurements, or digital photos of newsworthy events. Sites replicate collections as a whole unit to simplify indexing and access, and to address archivists' concerns that collections be kept contiguous (to simplify issues such as provenance). Here we treat all collections as equally worthy of preservation and equally difficult to preserve. If a collection is more important or more susceptible to failures than other collections, a site can simply try to make extra copies of that collection to improve its reliability. Sites should also take standard precautions, such as making off-line backups and protecting against viruses and hackers, to complement the reliable replication.

When a collection is replicated, the copies are kept accessible online, to enable periodic checks for corruption due to hardware or software failures. These checks can be conducted by both the archive site "owning" the collection and its replication partners. When corruption is discovered, the corrupted copy should be replaced as soon as possible by a new, pristine copy, so that there

are always several good copies of a collection. The process of check and repair is described in detail in [6]. If a site fails and does not recover, its partner sites must decide whether to take over responsibility for the site's collections, to ensure that they are not lost despite the failure of the site. If a site does not recover, another site can decide to take over stewardship of the failed site's collections. Since we are dealing with librarians and archivists, we believe they will have a motivation to ensure that the collections of permanently failed sites are preserved. Bid trading works to ensure that transient failures do not cause permanent loss.

A site (the "local site") with an important collection of size S will propose a trade to a remote site, requesting S bytes of space. If the remote site agrees, the two sites swap *deeds*, where a deed is the right of one site to use space at another site. Thus, the local site reserves some amount B of its space for use by the remote site, and the remote site reserves S bytes of its space for use by the local site. The local site can then use its deed for the remote site's space to make a copy of its collection at the remote site. Note that each site is agreeing to provide perpetual, online access to stored data, which means maintaining server machines, providing network connectivity, and so on, in addition to providing disk space. The remote site can hold on to its deed for the local site, or can use it to replicate a collection of its own. The local site will continue asking for trades until it has achieved its *replication goal* of G copies. A series of such binary, peer-to-peer trades between archives creates a trading network among many sites. Although this network is built up from local decisions made by sites, it serves a global purpose of preserving data through replication.

The trading negotiation must determine a "price" B for a trade: the amount of space that the local site must give to the remote site. In *fixed-price trading*, $S = B$, and the sites exchange equally sized deeds. A more general scheme is one in which B may be more or less than S, depending on the needs of the remote site. We can call this general scheme *bid trading*. An example is shown in Figure 2. Site A wishes to replicate a collection of size 80 GB. It calls an auction, announcing the auction size of $S = 80$ GB to the remote sites (solid lines in Figure 2a). Each site responds with a bid B_i (dashed lines in Figure 2a); this bid is the amount of space site A will have to give to make a trade. Site A chooses the winner as site D, which submitted the lowest bid $B_D = 65$ GB

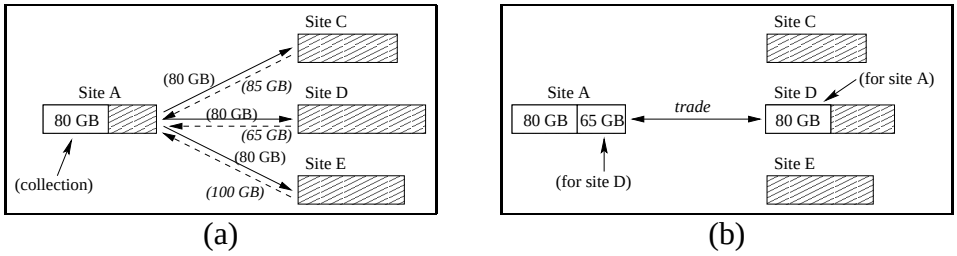

Figure 2: Bid trading example.

(although other criteria, such as the reliability of the site, may also be considered). Next a trade is conducted (Figure 2b), with sites A and D exchanging deeds. Now, site A can use its deed for site D to make a copy of its collection.

In this paper we examine how increasing the amount of freedom in the bidding system affects the resulting reliability. We can think of a "spectrum of freedom," illustrated by the following scenarios, ranging from the most restrictive (top of the list) to the least restrictive (bottom of list). (There are many other scenarios besides the ones we illustrate here.)

- *Fixed-Price Bids*. All sites follow the same fixed-price policy discussed above: A bid B must be the same as the amount of space requested, S.
- *Adaptive Bids.* All sites follow the same policy, but the policy takes into account local conditions. For example, the bid B may be determined by a function $f(R, S)$ that takes into account the available free space R at the site (and the requested space S).
- *Multiple Policies.* Sites are partitioned into classes, depending on factors such as their free space. For example, there would be a family of bidding functions $f_1, f_2, ...$, and all sites in a class use the same function.
- *Maverick Site.* We again have multiple classes, but now there is a single "maverick" site that follows its own policy. For example, a site may decide to implement its own version of the bidding software, and in so doing may not follow the policy for its class. This site is not malicious but simply follows a different auction or bid policy than other sites.
- *Free Market.* Each sites may use its own policy in an attempt to maximize its own benefit.

- *Malevolent Sites.* Some sites break the basic trading rules and try to subvert the system. For example, a site may promise to store a collection, and then delete it. Or a site could carry out a denial of service attack, generating so many message that other sites cannot trade.

In this paper we confine our attention to scenarios at the "restrictive" end of the spectrum, specifically the Fixed-Price, Adaptive, Multiple, and Maverick scenarios. The "permissive" scenarios have so many degrees of freedom that it is very hard to study them without first gaining an understanding of the more controlled scenarios. Furthermore, archival sites will almost certainly want to trade with known entities they trust, and could reasonably agree to a common price structure, as long as their autonomy is preserved. There might be some sites that due to implementation differences (or errors) do not follow this price structure exactly, and therefore it is interesting to study the Maverick scenario. However, since we are assuming trusted archival sites, we do not study here mechanisms that enforce the selected policies or rules, or that detect violations.

Such enforcement mechansims are necessary in the Free Market and Malevolent Sites scenarios, in which sites may try to game or subvert the system. For example, a site may try to gauge the current price levels in the system by calling many auctions and collecting many bids without any intention of actually making trades. Or, a site may accept as many bids at low prices as possible to try and gain a "monopoly" on the storage resources in the system. Game theoretic analysis may be useful to understand such behaviors and devise trading strategies to counteract them [3, 15]. In addition, the trading infrastructure must be made resilient to malicious sites that try to break the rules of fair exchange. For example, in our system, deeds serve as a bookkeeping mechanism among sites that trust each other. If Stanford and Georgia Tech trade deeds, it is unlikely that Georgia Tech will wantonly renege on its agreement. In the Free Market or Malevolent Sites scenarios, it is necessary to devise mechanisms to enforce deeds, for example using secure hashing of deeds [9], auditing [9, 25] or reputation systems [17]. Similarly, it is necessary to ensure that deeds are actually exchanged, for example to prevent a situation where one site collects a deed in a trade but then refuses to give a deed away. In general, dealing with behaviors in the Free Market and Malevolent Sites scenarios is a hard problem and beyond the scope of this paper.

2.1 Reliability

Our goal is to provide the most reliable storage for collections. Sites may fail (and lose data) with some probability, and we can measure the reliability with which a collection is stored by calculating the probability that the object is not lost despite site failures. For each site, we can calculate the local mean time to failure (MTTF), which is the expected time before any of that site's collections is lost. Our goal here is to find which policies guiding the decision making of a local site maximize the local data MTTF for that site.

For example, consider a network of three sites, A, B and C. Each of these sites could fail independently. For example, we can assume that over the course of some interval (say, one year), that a site has a ten percent chance of failure. This value reflects not only the reliability of the hardware that stores data, but also other factors such as bankruptcy, viruses, hackers, users who accidentally delete data, and so on.

If site A owns a collection, and stores a copy at site B, then the probability that the collection is lost permanently depends on the probability that both site A and site B fail at the same time. In a given year, this event has a probability of $P_F = 0.1 \times 0.1 = 0.01$. The MTTF of the collection is $1/P_F = 100$ years. If this is the only collection owned by site A, then we say that the MTTF for site A is 100 years. If site A makes another copy of its collection, to site C, P_F decreases to 0.001 and the MTTF for site A increases to 1000 years.

Note that our MTTF calculations assume that failures are repaired in the same year they occur, through a check and repair mechanism (described above). Thus, a loss of a collection occurs only when there are failures at all sites holding a collection copy in the same year. Certainly, the MTTF depends on the frequency of check and repair. However, our MTTF metric as defined gives us a good estimate of how reliable a policy is, especially when measured relative to other policies.

2.2 Trading process

When a site wishes to replicate a collection, it must either acquire a new deed for a remote site, or use an existing deed. In order to acquire a new deed, the local site calls an auction, inviting

remote sites to submit bids. The decision of when to call an auction is determined by the *auction calling policy* (see Section 3). An example of the steps that the auctioning site can take is shown in Appendix A. The auction procedure finds all the remote sites that do not already have a copy of collection C, and solicits bids from these sites. The auctioning site attempts to acquire S bytes from the bidding sites, where S is the size of collection C. It is possible that some (or all) of these sites will not bid (submitting a bid of $NULL$), either because they do not have as much space as the auctioning site is requesting, or because they do not want to trade at this time. If no sites bid (or if all remote sites already have a copy of C), then the auction terminates without any trading.

If at least one bid is submitted, then the auctioning site must pick a winner. Here, we assume the auctioning site picks the site that submitted the lowest bid B_W. If the auctioning site does not have enough space to satisfy the lowest bid, there is no winner and the auction will terminate. Alternatively, the auctioning site may choose to favor the most reliable bidding site, sites that have been dealt with before, or some combination of these and other factors.

Once a winner is chosen, then the sites trade. The auctioning site acquires a deed of size S, and must give the winning site a deed of size B_W (the winning site's bid). B_W may be more than, less than or the same as S. The auctioning site can use its new deed to store a copy of collection C.

When a local site is asked to bid in an auction, it calculates a bid B and send it to the auctioning site. The bidding site can choose a bid based on many factors, such as how urgently it needs to replicate its own collections, how scarce its local storage space is, how desirable it is to trade with the auctioning site, and so on. The policy that guides the construction of an appropriate bid is called the *bid policy*, and is described in Section 3. The simplest bid policy, always setting $B = S$, results in fixed-price trading.

3 Adaptive Bids scenario

In the Adaptive Bids scenario, all sites use the same global auction calling policy and the same global bid policy.

The *auction calling policy* is a set of rules for automatically deciding when to call an auction and for what collection. Here, we assume that sites call auctions when they need to make copies of their collections. If a site calls an auction and no remote sites bid (e.g. because the remote sites do not have enough storage space), it waits until the global state changes (e.g. another site joins or an existing site adds more storage) before calling another auction. We assume there is some mechanism for detecting this state change.

Once a site decides to call one or more auctions, it must decide which collections to replicate. The collections that must be most urgently replicated are those collections that are rarest (have the fewest copies). Thus, a site can call multiple auctions, one per collection, starting with the rarest collection. However, a site must decide how many collections to try and replicate during each round of auctions. It has two choices:

- *CallForAll*: call auctions for all of the collections. This policy tries to use the "call auction" mechanism to make as many copies as possible of each collection.
- *CallForRare*: call auctions only for the rarest collections. For example, a site may be trying to make G copies of every collection; G is a goal locally defined by the site administrator. We can define the "rare" collections as those that have less than $G/2$ copies, and the "abundant" collections as those that have at least $G/2$ copies. Rare collections are replicated when the local site calls an auction for them. Abundant collections can also be replicated, but only as a result of the local site bidding in an auction called by a remote site.

The *bid policy* is a set of rules for automatically calculating the bid for each auction. There is a huge space of possible bid policies. We cannot attempt to study them all, so we will restrict our examination to a subset of the possible policies. Specifically, we will examine a family of policies defined by two parameters: *I*, the width of the interval of potential bids, and *P()*, the *policy function* that determines how bids vary along this interval. We can call the bid policies described by these parameters *I-P policies*.

The policy function $P()$ reflects the local site's valuation of its own space resources versus resources at a site the local site wants to trade with in a particular auction. We specify that $0 \leq$

$P() \leq 1$. The interval I determines the maximum and minimum bids the local site will make in any auction. As an example, consider a policy where a site bids between $0.5 \times S$ and $1.5 \times S$ (where S is the amount of space the auctioning site is asking for). Then, the interval has width $I = 1$. Note that the interval is symmetric: the midpoint of the interval is at S. The bid policy may dictate that sites bid low when their local storage space is abundant, and bid high when their storage space is more scarce. In this case, $P() \propto U$, where U is the fraction of local storage space that has been used up.

Formally, a site can calculate its bid B as

$$B = S \times (I \times P() + (1 - I/2)) \tag{1}$$

We specify that $0 \leq I \leq 2$. If $I < 0$ or $I > 2$ then Equation 1 can produce negative bids, which are not allowed in our framework. If the interval width $I = 0$, fixed-price trading results.

Here, we examine bid policies with different values of I and $P()$. We have studied four different policy functions $P()$, which give us four different bid policies: FreeSpace, UsedSpace, AbundantCollection and RareCollection. Recall that under the Adaptive Bid scenario we are studying here, all sites would agree to use one of the following options:

FreeSpace: A site bids more when it has more free space. In this case, $P() = K/T$, where K is the amount of free local space, and T is the total amount of local space (used and free). Under the FreeSpace policy, a site tends to win auctions when its space is scarce, because then the site bids low. This may be the best policy since space scarcity makes trading more difficult, and thus sites should try to win as many auctions as possible.

UsedSpace: A site bids more when more of its space is used. $P() = (T - K)/T$. Under this policy, sites tend to bid low and win auctions when their space is abundant, but bid high (and lose more auctions) when their space is scarce. This policy may be preferred to allow sites to hoard local space when that space is scarce.

AbundantCollection: A site bids more when its collections are abundant. If C is the number

of copies of the rarest collection (the collection with the fewest copies), and G is a "goal" number of copies to make of each collection, then $P() = C/G$. In other words, when there are very few copies of the rarest collection, then the site bids low, wins auctions, and replicates its rare collections. When there are many copies of its rarest collection (and thus many copies of every collection), the site bids higher, and wins few auctions. This policy may be preferred because it allows sites to make more trades when their collections are rare. In order to keep $P()$ between 0 and 1, we treat $C/G > 1$ as 1.

RareCollection: A site bids more when its collections are rare. In this case, $P() = (G - C)/G$. In order to keep $P()$ between 0 and 1, we treat $G - C < 0$ as 0. Although a site will bid high and win fewer auctions when its collections are rare, each time it wins an auction the site will acquire a large amount of space at the auctioning site. This will allow sites to replicate many collections when they win auctions.

In previous work [7, 8], we have examined the Fixed-Price Bids scenario. This scheme is even more restrictive than the Adaptive Bids scenario, since sites cannot bid at all. In Section 6, we compare the reliability achievable under bid trading to that achievable under fixed-price bidding.

4 Multiple Policies scenario

Different sites have different resources and resource requirements, and it may be that there is no one policy that is good for all sites. In the Multiple Policies scenario, we partition the sites into distinct classes, and allow each class to use a different policy. For example, we may create a class of sites that have an initially large amount of storage space, and another class of sites that have less storage space. The sites in the high capacity class could use a policy that best utilizes their abundant resources, while the low capacity sites would use a policy that best manages their scarce resources. For the Multiple Policies scenario, we can study the same alternatives outlined in Section 3. In other words, once we define the classes of sites, we can determine the auction call policy and bid policy that provides the best reliability for each class.

5 Maverick Site scenario

The data trading network is founded on a principle of collective benefit from individual action. Sites seek to help themselves, and in doing so, help other sites. Thus, it is reasonable for sites to use a common bidding structure to maximize the benefit for everybody. However, it is possible that due to programming differences, some sites may not implement the same policies as other sites. Such sites are not "malicious," but simply use different policies due to implementation decisions (or implementation bugs). In the Maverick Site scenario, most sites use the policies that are best for their class, but one site deviates from these policies.

Although there are a large number of ways in which a maverick site may act differently than other sites, we can only examine a few here. Specifically, we will examine some variations of the policies for calling auctions and choosing bids that a maverick site may implement. In Section 6.4 we examine the effect (if any) these behaviors have on the reliability of sites in the trading network.

5.1 Maverick auction calling behaviors

Trades occur when auctions are called. Although sites can choose to call or not call an auction depending on their local situations, there are behaviors that differ markedly from either the CallForAll or CallForRare policies. Here, we study the *AlwaysCall* behavior, and the *NeverCall* behavior.

AlwaysCall: a site calls auctions constantly. A local site may do this to reserve a large amount of space at remote sites for its own use. In every auction, the local site must give some of its own space to the winner of the auction. Thus, we would expect this behavior to only benefit sites that have a lot of local space to give away. Otherwise, the maverick site could call lots of auctions but would only be able to conduct a trade as a result of a few auctions. This behavior would not normally be followed by all sites because sites are expected to call auctions when they need to make a trade, not simply because they wish to reserve space.

NeverCall: a site never calls auctions. A local site may prefer not to implement the machinery needed to conduct auctions, or may simply prefer to set the price of all trades it participates in. This

behavior would not normally be followed by all sites because sites are expected to call auctions; otherwise, no trading would ever occur.

5.2 Maverick bidding behaviors

Under normal bid policies, sites sometimes bid high ($B > S$) and sometimes bid low ($B < S$). However, maverick sites may choose instead to follow a simpler policy. Here, we study the *BidHigh*, *BidLow* and *NeverBid* behaviors.

BidHigh: a site consistently bids high; $B > S$ always. A maverick site may decide to do this so that whenever it wins an auction, it receives a lot of remote space while giving away relatively little local space. Such a site is analogous to a "bargain hunter" that only buys if the price is good. Normally sites would bid based on their resource situation (sometimes bidding high and sometimes low), not just to get bargains always.

BidLow: a site consistently bids low, e.g. $B < S$ always. The benefit of BidLow is that a site wins more auctions, and thus reserves more space at remote sites for its own use (even though it may pay a lot for the storage). If every site bid low, then no one site would consistently win auctions. In other words, normal sites win some auctions and lose some auctions, but a maverick site tries to win every auction.

NeverBid: a site never submits a bid to an auction. The site can still conduct trading, but does so only by calling auctions. A site may try to do this so that it can always determine when a trade occurs, and never has to wait for a remote site to call an auction. The trading network assumes sites bid in auctions; if all sites refused to bid in auctions then the network would fall apart as no auctions would result in trades.

6 Results

We have conducted a series of experiments to study the tradeoffs involved in bid trading. In these experiments, we conducted simulated trading sessions between archive sites, comparing various

Variable	*Description*	*Base values*
S	Number of sites	10 to 15
F	Site storage factor	2 to 6
P	Site reliability	0.9
$CperS_{MIN}$, $CperS_{MAX}$	Min/max collections per site	$CperS_{MIN} = 4$, $CperS_{MAX} = 25$
$Csize_{MIN}$, $Csize_{MAX}$	Min/max collection size	$Csize_{MIN} = 50$ GB, $Csize_{MAX} = 1000$ GB
$Ctot_{MIN}$, $Ctot_{MAX}$	Min/max total data per site	$Ctot_{MIN} = 200$ GB, $Ctot_{MAX} = 10,000$ GB
G_M	Minimum replication goal	3 copies
G	Ideal replication goal	6 copies

Table 1: Simulation variables.

bid and auction calling policies under the Adaptive Bids, Multiple Policies and Maverick Sites scenarios. In this section, we discuss our simulator, and present the results of our experiments.

6.1 The bid trading simulator

Our simulator conducts a series of simulated auctions and trades, and the resulting local data reliabilities are then calculated. Table 1 lists the key variables in the simulation and the values we used; these variables are described below.

The simulator generates a *trading scenario*, which contains a set of sites, each of which has a quantity of archival storage space. The number of collections "owned" by the site is randomly chosen between $CperS_{MIN}$ and $CperS_{MAX}$. Each scenario has $10 \leq S \leq 15$ sites. Our experiments indicate that although much larger networks are feasible, a larger number of sites is not necessary to achieve high reliability; see [7]. Collections "appear" in globally random order; this models collections being created and archived over time. A site is "born" when the first of its collections is archived, and no site has advance knowledge about the creation of other sites or collections. Collections all have different, randomly chosen sizes between $Csize_{MIN}$ and $Csize_{MAX}$. The sum of the sizes of all of the collections assigned to a site ranges from $Ctot_{MIN}$ to $Ctot_{MAX}$. The values we chose for these variables represent a highly diverse trading network with small and large collections and sites with small or large amounts of data. The archival storage space assigned to the site is the *storage factor* F of the site multiplied by the $Ctot$ at the site. This models a situation where a site administrator chooses to install F times as much disk space as needed to store the

locally owned collections. The space left after storing collections is *public* space used to store copies of collections owned by other sites. In each scenario, some sites may have a large F (e.g. 6) while others may have a small F (e.g. 2). Site i fails with probability $R_i = 0.1$, as described in Section 2.1. (For experiments where the site reliability differs, but specifically in the context of fixed-price trading, see [7].)

Sites try to make at least G_M copies of each collection. Some bid policies (i.e. AbundantCollection and RareCollection) depend on G, which is the ideal goal number of copies. We assume here that all sites use the same G and G_M values. In general, sites may assign higher G and G_M values to more important collections. We do not expect this to significantly alter the trading dynamic, as there is not much difference between trying to make three copies of two different collections and six copies of one collection.

In the following sections, we examine the improvement or detriment due to using one policy versus another. For example, if a site achieves a MTTF of 100 years using policy X, and a MTTF of 300 years using policy Y, we would report a 200 percent improvement for using policy Y versus a baseline of policy X. For each experiment, we ran 1200 simulations, and used the standard deviation of our measurements to calculate 95 percent confidence intervals. In our experiments, these intervals were ± 50 or less except where noted. For example, the average percent MTTF improvement for policy Y (versus policy X) might be 200 ± 50 (with 95 percent confidence).

6.2 Adaptive Bids scenario

First, we examined which policies resulted in the highest reliability under the Adaptive Bids scenario, where all sites use the same policy. We studied both the auction policy, which describes when a site calls an auction, and the bid policy, which determines how a site bids in an auction.

6.2.1 Auction policies

The auction policy dictates when a site will call an auction, and for which collection. With the CallForAll policy, a local site repeatedly calls auctions for each of its collections, in rarest first

order, as long as the local site is receiving bids from remote sites. The CallForRare policy is the same, except that the local site does not call auctions for collections with at least $G_M = 3$ copies.

We ran a set of experiments where we tried both auction calling policies with each bid policy (including the FixedPrice policy). Our results (not shown) indicated that the CallForRare policy was always better. For example, when sites used the UsedSpace bid policy, the CallForRare auction policy provided up to 850 ± 100 percent improvement in MTTF over the CallForAll auction policy (when $F = 5.8$). The results for other bid policies are similar: CallForRare was better than CallForAll regardless of which bid policy was used. These results suggest that it is detrimental to reliability if a site calls too many auctions. Although the CallForAll policy caused the site to actively try to replicate collections by calling auctions, sites called too many auctions too soon, using up their local storage, and too few copies were made of collections deposited later in the trading session. Instead, sites should try to strike a balance between calling auctions themselves (to ensure that collections are replicated a few times), and bidding in auctions called by other sites (to replicate the collections further). This is what happened with the CallForRare policy.

6.2.2 Bid policies

Next, we examined different bid policies. The bid policies described in Section 3 were implemented by calculating B using Equation 1. To aid the reader, we review the policies here:

- *FreeSpace*: A site bid more when it had more free space.
- *UsedSpace*: A site bid more when it had used up more of its space.
- *AbundantCollection*: A site bid more when it had already made many copies of its collections.
- *Rare Collection*: A site bid more when there were few copies made of its collections.

If multiple sites submitted identical minimum bids, the local site chose the site with which it had traded the most in the past. If this did not break the tie, the local site chose randomly among the tied sites. (Choosing previous partners first produces higher reliability than making random the first tiebreaker; see [8].) For each policy, $I = 1$. We also tested the FixedPrice policy (e.g. $I = 0$

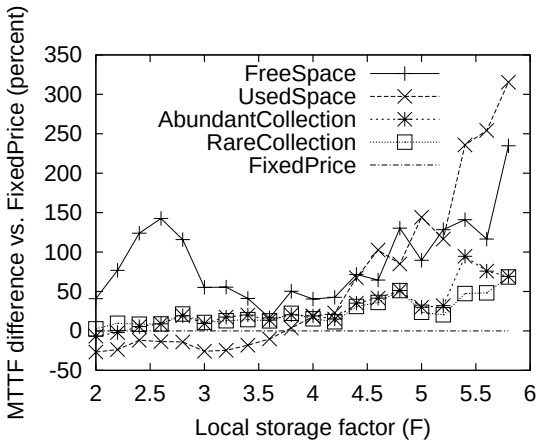

Figure 3: Bid policies in the Adaptive Bids scenario.

and there is no bidding). Comparing against the FixedPrice policy allows us to determine whether bid trading is beneficial versus a non-bidding data trading network.

The results are shown in Figure 3, which shows the percent MTTF change for each bid policy versus a baseline of the FixedPrice policy. The horizontal axis in this figure is F: the ratio of the total storage space at a site to the amount of locally owned data at that site. The figure shows that no one policy was best. For high capacity sites (with $F \geq 4.4$), either the UsedSpace policy or FreeSpace policy was best. For these policies, the 95 percent confidence interval is ± 50 for $F < 5.6$ and ± 100 for $F \geq 5.6$; thus for high capacity sites the confidence intervals for the UsedSpace and FreeSpace policies overlapped and neither was statistically "better" than the other. (Also, the dips in peaks for UsedSpace and FreeSpace for $F > 4.4$ are noise within the confidence interval.) For mid capacity sites ($3.2 \leq F < 4.4$) all policies were roughly the same, since all are within the confidence interval of ± 50. For low capacity sites ($F < 3.2$), all policies were the same (within the confidence interval) as the FixedPrice baseline, except FreeSpace, which provides up to 140 percent improvement over FixedPrice.

The results for the FreeSpace policy are due to two competing effects: sites bid low and won many auctions, or bid high and won big in a few auctions. Low capacity sites ($F < 2.6$), which often cannot bid at all, benefited from FreeSpace. When low capacity sites did bid, they tended to have little free space and thus bid low, winning frequently. In the range $2.6 \leq F < 4.4$, sites still

could not bid in very many auctions, and tended to lose the ones they did bid in since they were bidding comparatively higher (since they have more free space). For $F \geq 4.4$, the "winning big" effect dominated, since high capacity sites could bid in many auctions and had a higher chance of participating in an auction they could win with a high bid.

Under the UsedSpace policy, sites bid more when they have little free space. In this case, high capacity sites (which usually have lots of free space) bid low and won auctions. Winning many auctions (even with low bids) resulted in a large total amount of remote space. Low capacity sites won fewer auctions under UsedSpace because they were bidding higher. As noted above, low capacity sites only benefit by bidding low which they could not do under UsedSpace.

This experiment suggests that it may be better if high capacity sites and low capacity sites use different policies. This is the Multiple Policies scenario, which we study next.

6.3 Multiple Policies scenario

In the Multiple Policies scenario different sites use different policies based on some partition of the sites. The results in Section 6.2.1 suggest that all sites benefited from the CallForRare policy, so we did not study the case where different classes used different auction policies. However, Figure 3 suggests that for bid policies, the storage factor F is a good way to partition sites into classes. Therefore, we constructed three classes: *high capacity* sites ($F \geq 4.4$), *mid capacity* sites ($3.2 \leq F < 4.4$) and *low capacity* sites ($F < 3.2$).

We ran simulations in which all of the sites in each class used the same bid policy, and we varied the policies used by different classes. This resulted in simulating $5^3 = 125$ combinations of bid policies (since there are five policies and three classes.)

Our results indicate first that high capacity sites ($F > 3.4$) should use the UsedSpace policy. UsedSpace allowed high capacity sites to bid low and win many auctions, so the sites could make as many copies as possible of their collections. In fact, UsedSpace is best for high capacity sites regardless of the policy used by mid or low capacity sites.

Second, low capacity sites ($F \leq 3.4$) should use the FreeSpace policy. FreeSpace allowed

low capacity sites to bid low and win many of the auctions they participated in, so the sites could aggressively try to make at least 3 copies of their collections. Again, FreeSpace was best for low capacity sites regardless of the policy used by other sites.

Third, there is not one best policy for mid capacity sites. Instead, sites with $F > 3.4$ benefit most from UsedSpace, while sites with $F \leq 3.4$ benefit most from FreeSpace. This indicates that the the best class division is actually two classes, with low capacity $F \leq 3.4$ and high capacity $F > 3.4$, rather than three classes. The dividing point comes from the fact that sites are trying to make at least $G_M = 3$ copies. This means a site with $F > 3.4$ had enough space to make 3 copies, and intuitively had a high storage capacity relative to the storage needed to make trades. A site with $F \leq 3.4$ has trouble making 3 copies, and intuitively had low storage capacity relative to the needed storage. In further experiments with two classes, high capacity sites ($F > 3.4$) did best with UsedSpace and low capacity sites ($F \leq 3.4$) did best with FreeSpace.

In order to examine the impact of the bid interval width I on reliability, we ran an experiment where I varied between 0 and 2 for high capacity ($F > 3.4$) sites using UsedSpace, while low capacity sites ($F \leq 3.4$) used FreeSpace (with $I = 1$). (We tested only $0 \leq I \leq 2$ since if $I > 2$ then the minimum bid is negative; see Section 3.) The results for high capacity sites indicate that although sites achieved better reliability with $I > 0$, with up to 100 percent improvement in MTTF versus $I = 0$, the actual value of I did not significantly impact MTTF.

We also ran an experiment where I varied between 0 and 2 for low capacity sites using FreeSpace, while high capacity sites used UsedSpace with $I = 1$. The results indicate that low capacity sites achieved the highest reliability with $I = 2$, with up to a 420 percent improvement over $I = 0$. By increasing the bid span, low capacity sites magnified the benefits of the free space policy: they won even more auctions, by bidding lower more often.

6.4 Maverick Site scenario

Some sites may decide not to follow the best policy for their storage class as a whole. Instead, they may behave differently, due to implementation differences or bugs. This situation is the Maverick

Site scenario. Here, we consider whether the behaviors outlined in Section 5 help or hurt the trading process. Specifically, we study two questions in this section:

- Does a maverick site achieve a higher reliability than the other sites in its class?
- Does the differing behavior reduce the reliability achieved by the sites that are following their class's policy?

We have implemented maverick behaviors as described in Section 5. With the BidHigh behavior, the maverick site used $B = 1.5 \times S$ always, and with the BidLow behavior, the maverick site used $B = 0.5 \times S$ always. With the AlwaysCall behavior, the maverick site continually called auctions of size 50 GB (the minimum collection size in our simulations) in addition to regular auctions for its collections. With the NeverCall and NeverBid behaviors, the maverick site never called auctions and never bid in auctions (respectively). While this is not an exhaustive list of the behaviors sites may engage in, they represent a variety of ways in which sites may behave differently than the rest of the sites in their class. In that sense, studying these behaviors helps us to get an idea of the effects of implementation differences among peers. We report here results for a single maverick site in a given trading network. We have also run experiments with two and three maverick sites; the results (not shown) differ in the absolute reliabilities achieved but our general conclusions remain the same.

The results were:

- A maverick high capacity site sometimes achieves higher reliability than other high capacity sites from the BidHigh behavior, but does not harm other sites doing so.
- A maverick high capacity site may also benefit from the NeverCall policy, and in doing so may harm other sites.
- A maverick low capacity site experiences lower reliability than the other sites in its class.

We first examined the situation where a maverick high capacity site deviated from the behavior recommended for its class. In this case, during each simulation, a high capacity site was chosen randomly as the "maverick" site. The results for the BidHigh behavior is shown in Figure 4a. This

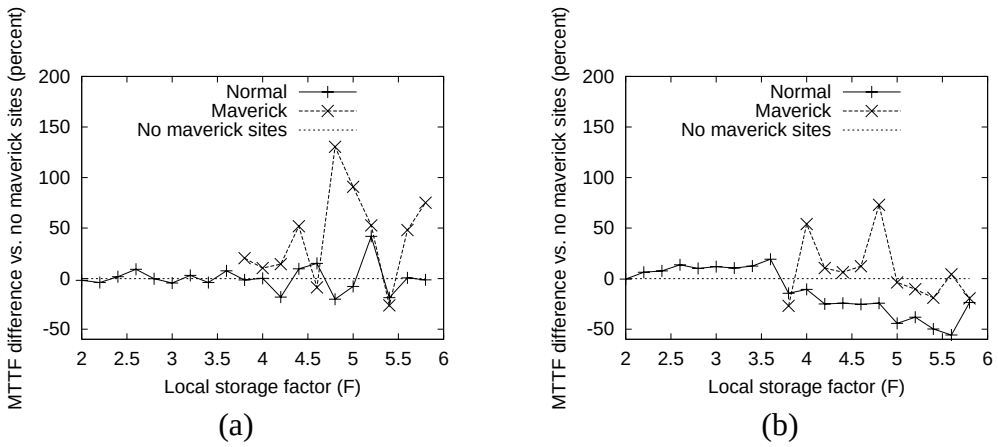

Figure 4: Maverick behaviors for high capacity sites: (a) BidHigh and (b) NeverCall.

figure shows two curves: one curve for the maverick site (labeled "Maverick") and one curve for the other sites in the same simulations as the maverick site but that themselves are not deviating (labeled "Normal"). Both of these curves represent the percent change in reliability for a site versus a baseline of no maverick sites (e.g., the Multiple Policies scenario).

The figure shows that the MTTF difference for maverick sites varied widely, sometimes with an increase in MTTF (up to 130 percent) and sometimes with a decrease (by up to 25 percent) versus the case where the site did not deviate. Moreover, the variance in our measurements was very high: the 95 percent confidence interval for the "Maverick" curve was ± 175 percentage points. The result is that the average plotted in the figure is very noisy with many dips and peaks within the wide confidence interval; for any given F a Maverick site may have experienced a large benefit or detriment. In order to understand this variability, we must understand the situations in which the BidHigh behavior is beneficial. BidHigh helps the maverick site because the site is able to get a large amount of space at the remote site, while giving away comparatively little. On the other hand, a BidHigh site may not win very many auctions, since it is bidding higher than other sites, and low bidders win an auction. In some trading sessions, the maverick site was frequently the lone bidder in an auction, and thus acquired a large amount of remote space at little cost to itself. In other sessions, there were usually more bidders in an auction, and thus the maverick site won few auctions, made fewer trades and experienced a loss in reliability. The end result is that the BidHigh behavior is "risky;" the maverick site only sometimes achieved high reliability.

However, Figure 4a also indicates that non-maverick sites did not experience a significant decrease in reliability versus the case where no site is maverick. (The dips and peaks in the "Normal" curve are noise within the 95 percent confidence interval of ± 50 for $F < 5.2$ and ± 75 for $F \geq 5.2$.) Although the maverick site was able to extract a high price in an auction, other sites were still able to make copies of their collections and achieve reliability. This indicates that the BidHigh behavior is not likely to decrease the reliability of the system.

Figure 4b shows the results from another experiment, where one high capacity site pursued the NeverCall behavior. As with the BidHigh behavior, the maverick site sometimes did well (achieving up to a 75 percent increase in MTTF) and sometimes did poorly (achieving up to a 25 percent decrease in MTTF). Once again, the variance was very high: the confidence interval for the "Maverick" curve is ± 100, resulting in a noisy average with many dips and peaks within this interval. Recall that a high capacity site used the UsedSpace policy, often bidding low and winning auctions. When the site's storage space began to fill up, the site started losing auctions, because it was bidding higher. Normally in this situation, a site still trades by calling auctions. However, a maverick site refused to call auctions, instead bidding (and bidding high). If the maverick site was the only bidder, then it got a large amount of remote space and made several copies of its collections. If there were other bidders, the maverick site lost auctions and made no trades. Thus, as with the BidHigh behavior, sometimes NeverCall benefited a maverick site and sometimes hurt the site. This produced the high variance observed in our results.

There is a difference in the case of NeverCall, however: non-maverick sites may be hurt by this behavior. This is because the maverick site either won auctions at high prices and reserved much of the space in the system for itself, or lost auctions and therefore did not give away its own space. In either case, some sites that may otherwise have used this space could not, resulting in less reliability for those sites. This is seen most clearly in Figure 4b in the case of $F = 5.6$, where the decrease in MTTF of 56 percent verses "No maverick sites" is larger than the ± 50 confidence interval. Non-maverick sites may need to pursue some corrective to discourage sites from following the NeverCall behavior. For example, they can attempt to identify a maverick site

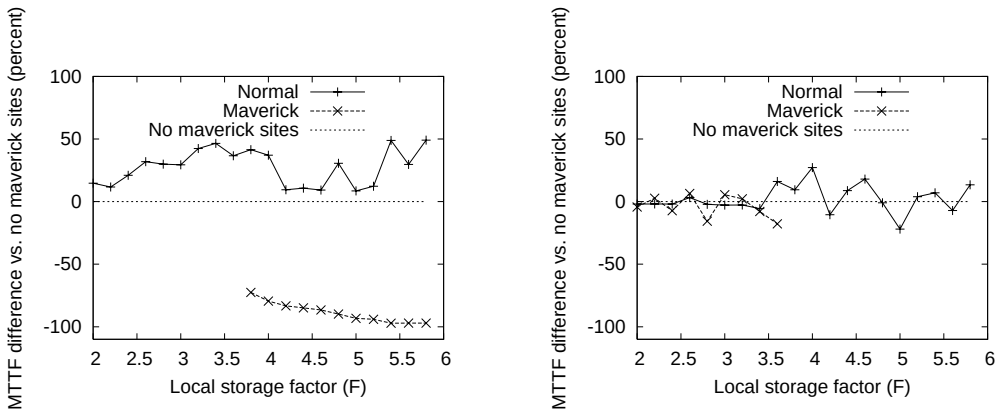

Figure 5: Maverick behaviors: (a) BidLow, high capacity site and (b) BidHigh, low capacity site.

ask it to change its implementation, or refuse to trade with it altogether, encouraging the maverick site to use a standard policy.

Our experiments have also shown that other maverick behaviors result in sharply reduced reliability for the maverick site. To illustrate, the results for BidLow are shown in Figure 5a. In this case, a high capacity maverick site consistently bid low in all of the auctions it participated in. As the figure shows, BidLow decreased reliability: although the maverick site won many auctions, it always did so by giving away much of its own space and getting little in return. NeverBid and AlwaysCall (results not shown) also hurt sites. This is because the local site was at the mercy of bids cast by other sites: most of the site's trades (under AlwaysCall) or all of the sites trades (under NeverBid) come when it is the auctioner. By losing price-setting flexibility, a maverick site loses the benefits that bid trading gives to sites. Moreover, in the case of AlwaysCall, a site may acquire many deeds at many sites, but there is no guarantee it will acquire a large enough deed at any site to be useful. As a consequence, the site used up all of its local space without necessarily replicating many of its collections.

In no case does a maverick behavior benefit a low capacity site. For example, the results for a low capacity maverick site using BidHigh is shown in Figure 5b. The results for other behaviors with low capacity sites are similar and not shown. Low capacity sites are rarely the only bidder in auctions, because their lack of storage space means that they often cannot bid at all. As noted above, being the only bidder is key to benefiting from the BidHigh or NeverCall behaviors.

7 Related work

A common approach to managing failures is to make local backups of important data. Unfortunately, a catastrophic site-wide failure can damage both the primary and backup copies. Moreover, over very long time spans, local backup media can decay and lose data. Remote backups provide protection against catastrophic failures. However, to protect against failures in the long term, the data must be periodically checked, and it is expensive to load and check remote backups (which are typically stored as tape archives). Our goal is to minimize the cost to libraries, so they can spend their money on content, not infrastructure.

Another approach is online replication, such as replicated DBMS's [2, 16], replicated filesystems [18, 21] and RAID disk arrays [26]. Such schemes utilize replication to protect against failures in the short term. However, they do not provide a high level of autonomy to the nodes participating in the replication network, relying instead on a central controller to determine data placement or manage free-space tables. Also, traditional solutions are concerned with load distribution, query time and update performance, in addition to reliability [11, 30, 35]. Thus, traditional replicated databases tend to trade some reliability for increased performance [20]. Here, we are primarily concerned about preservation (given the constraint of preserving site autonomy). Finally, traditional replicated schemes typically require the purchase of new hardware. One of our goals is to utilize existing infrastructure whenever possible.

Systems such as the Archival Intermemory [4] and OceanStore [19] preserve digital objects but do not give sites control over where their data is placed (or what data they store). These systems could use the techniques we describe here to determine where data will be replicated while preserving site autonomy. Our work is also related to existing peer to peer trading systems such as FreeHaven [10] or Gnutella. Systems like Gnutella provide searching but do not guarantee preservation. A search facility could be built on the system we describe here. FreeHaven guarantees preservation but also focuses on anonymity, which requires different constraints and techniques than in our domain. Samsara [9] extends our concept of storage deeds [8, 7, 5] (which they call "claims") to ensure fair exchanges.

Auction theory has been extensively developed in both economics and computer science. Many auction theory results are theorems about optimal allocation in abstract models, and [12] notes that work is needed to apply theoretical mechanisms to real systems. Moreover, auction theorists usually make assumptions that are not applicable here, such as the existence of a currency different from the resources themselves, a distinction between producers and consumers, and global pricing [33]. Others have looked at "efficient clearing": the best way to assign resources to bidders so as to maximize utility across the system, assuming all resources and bids are known at the same time [1]. In our system, resources and bids appear over time, and archives, which make copies as soon as possible to avoid failures, cannot wait until all resources and bids are known.

It has been shown that under certain conditions, second-price auctions have certain desirable properties, including efficient allocation and the fact that the dominant strategy for bidders is to declare the true value of a resource as the bid. [23]. However, second-price auction systems are rare in practice [22, 28, 29], usually because the theoretical conditions do not appear in real systems. For example, one condition is that bidders are risk-neutral, and particularly that not winning an auction is an acceptable outcome. In our system, not winning an auction is often a bad thing (since that means that a collection may not be replicated), and thus bidders are likely to be averse to the risk of losing an auction they would prefer to win. With risk-averse bidders, both bidders and auctioners can achieve higher utility under the first-price model [28, 29]. This, along with other reasons outlined in [28], as well as the fact that first price auctions fit more naturally into our bartering model, is why we have chosen to use first-price auctions.

Several systems have attempted to apply market-oriented programming, and specifically auction techniques, to resource allocation problems. Schwartz and Kraus [31] survey methods for using auctions to distribute data collections. They assume that there is a common currency, that there is one copy of each collection, and that the performance metric is access time. Some or all of these assumptions are shared by computational economies such as the Blue-Skies digital library [24], the Mariposa transaction processing system [32], and Ferguson, Nikolaou and Yemini's replicated data processing economy [13]. Our unique application, replication to achieve reliability,

means that we can both draw from and extend this previous work.

Others have investigated incentive mechanisms besides auctions for encouraging peers to contribute resources. For example, Golle, Leyton-Brown and Mironov [15] describe using currency-based micropayments to pay for resources exchanges. While their mechanism has nice game-theoretical properties, we believe that it is not the best match for a digital preservation system. One reason is that it relies on a currency, which can become unstable over time (as observed in other market-oriented programming systems [34]) and requires a central accounting server that can become a single point of failure. Moreover, digital archives are motivated by the need for preservation, not currency-based profit. Buragohain, Agrawal and Suri [3] use differentiated quality of service to incentivize peers to contribute resources. Again, despite attractive game-theoretical equilibria, this system is inherently focused on short time spans, where differentiated services architectures are appropriate. Over the long term, only two levels of service matter for a collection: successful preservation or loss. Our bid trading mechanism allows peers to exchange resources without relying on a potentially unstable currency or short term QoS incentives.

8 Conclusion

Bid trading is a mechanism we have developed for allowing sites to conduct peer-to-peer data trading to achieve high reliability. In this paper, we have examined the bid trading paradigm in detail. We have described the bid trading process, which allows a local site to determine how much space at the remote site to ask for in return for giving a deed of a certain size to the remote site. We have described how the auction and bidding process works, and examined policies for deciding when to call an auction and how much to bid. Using a trading simulator, we have determined which policies provide the highest reliability: the UsedSpace policy for sites with a lot of storage capacity, and the FreeSpace policy for lesser capacity sites. Our experiments also show that several maverick behaviors do not significantly harm the system. Our results show that bid trading is an effective model for peer-to-peer data trading and preservation for a cooperating network of sites.

We have focused on a network of sites that know and generally trust each other. Such sites can agree on a general policy framework, perhaps by using a common implementation of the trading agents. In more permissive scenarios, sites may not have the same level of trust or cooperation. In the Free Market scenario, sites may pursue any strategy (and even change strategies) to maximize their benefit. For example, a site may call auctions simply to determine what other sites are willing to bid, but then not accept any trades. Game theoretic analysis may be useful to examine the strategies that sites can adopt to ensure high reliability in this scenario. Moreover, in the Malevolent Sites scenario, sites may actually break the rules of the system. For example, a site may accept a deed but then refuse to give one in return, or may renege on the terms of a deed. In this case, secure signatures, auditing and reputation mechanisms may be needed to detect and punish malicious sites. These scenarios can form interesting avenues for further exploration of bid trading.

References

[1] A. Andersson, M. Tenhunen, and F. Ygge. Integer programming for combinatorial auction winner determination. In *Proc. Int. Conf. on Multi-Agent Systems*, July 2000.

[2] F. B. Bastani and I-Ling Yen. A fault tolerant replicated storage system. In *Proc. ICDE*, May 1987.

[3] C. Buragohain, D. Agrawal, and S. Suri. A game theoretic framework for incentives in P2P systems. In *Proc. of the Third International Conference on Peer-to-Peer Computing (P2P)*, September 2003.

[4] Y. Chen, J. Edler, A. Goldberg, A. Gottlieb, S. Sobti, and P. N. Yianilos. A prototype implementation of archival intermemory. In *Proc. ACM Int'l Conf. on Digital Libraries*, 1999.

[5] B. F. Cooper and H. Garcia-Molina. Bidding for storage space in a peer-to-peer data preservation system. In *Proc. Int. Conf. on Distributed Computing Systems (ICDCS)*, July 2002.

[6] B.F. Cooper, A. Crespo, and H. Garcia-Molina. Implementing a reliable digital object archive. In *Proc. European Conf. on Digital Libraries (ECDL)*, Sept. 2000. In LNCS (Springer-Verlag) volume 1923.

[7] B.F. Cooper and H. Garcia-Molina. Creating trading networks of digital archives. In *Proc. 1st Joint ACM/IEEE Conference on Digital Libraries (JCDL)*, June 2001.

[8] B.F. Cooper and H. Garcia-Molina. Peer-to-peer data trading to preserve information. *ACM Transactions on Information Systems*, 20(2), Apr. 2002.

[9] L. P. Cox and B. D. Noble. Samsara: Honor among thieves in peer-to-peer storage. In *Symposium on Operating System Principles (SOSP)*, Oct. 2003.

[10] R. Dingledine, M.J. Freedman, and D. Molnar. The FreeHaven Project: Distributed anonymous storage service. In *Proc. of the Workshop on Design Issues in Anonymity and Unobservability*, July 2000.

[11] X. Du and F. Maryanski. Data allocation in a dynamically reconfigurable environment. In *Proc. ICDE*, Feb. 1988.

[12] D. Ferguson, C. Nikolaou, J. Sairamesh, and Y. Yemini. Economic models for allocating resources in computer systems. *Market-Based Control: A Paradigm for Distributed Resource Allocation*, 1996.

[13] D. Ferguson, C. Nikolaou, and Y. Yemini. An economy for managing replicated data in autonomous decentralized systems. In *Proc. Int. Conf. Symp. on Autonomous Decentralized Sys.*, Mar. 1993.

[14] J. Garrett and D. Waters. Preserving digital information: Report of the Task Force on Archiving of Digital Information, May 1996. Accessible at http://www.rlg.org/ArchTF/.

[15] P. Golle, K. Leyton-Brown, and I. Mironov. Incentives for sharing in peer-to-peer networks. In *Proc. of the 3rd ACM conference on Electronic Commerce*, October 2001.

[16] J. Gray, P. Helland, P. O'Neal, and D. Shasha. The dangers of replication and a solution. In *Proc. SIGMOD*, June 1996.

[17] S. Kamvar, M.T. Schlosser, and H. Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *Proc. of the World Wide Web Conference*, May 2003.

[18] J. J. Kistler and M. Satyanarayanan. Disconnected operation in the Coda file system. *ACM TOCS*, 10(1):3–25, Feb. 1992.

[19] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. OceanStore: An architecture for global-scale persistent storage. In *Proc. ASPLOS*, Nov. 2000.

[20] E. Lee and C. Thekkath. Petal: Distributed virtual disks. In *Proc. 7th ASPLOS*, Oct. 1996.

[21] B. Liskov, S. Ghemawat, R. Gruber, P. Johnson, L. Shrira, and M. Williams. Replication in the Harp file system. In *Proc. 13th SOSP*, Oct. 1991.

[22] R. P. McAfee and J. McMillan. Auctions and bidding. *J. of Economic Literature*, 25(2):699–738, June 1987.

[23] P. Milgrom. Auctions and bidding: A primer. *J. of Economic Perspectives*, 3(3):3–22, Summer 1989.

[24] T. Mullen and M. Wellman. A simple computational market for network information services. In *Proc. Int. Conference on Multi-Agent Systems*, June 1995.

[25] T.W. Ngan, D. Wallach, and P. Druschel. Enforcing fair sharing of peer-to-peer resources. In *Proc. of the 2nd International Workshop on Peer-to-Peer Systems*, February 2003.

[26] D. Patterson, G. Gibson, and R. H. Katz. A case for redundant arrays of inexpensive disks (RAID). *SIGMOD Record*, 17(3):109–116, September 1988.

[27] D. S. H. Rosenthal and V. Reich. Permanent web publishing. In *Proc. 2000 USENIX Annual Technical Conference*, June 2000.

[28] M. H. Rothkopf, T. J. Teisberg, and E. P. Kahn. Why are Vickrey auctions rare? *The Journal of Political Economy*, 98(1):94–109, February 1990.

[29] T. Sandholm. Limitations of the Vickrey auction in computational multiagent systems. In *Proceedings of the International Conference on Multiagent Systems (ICMAS)*, 1996.

[30] H. Sandhu and S. Zhou. Cluster-based file replication in large-scale distributed systems. In *Proc. SIGMETRICS*, June 1992.

[31] R. Schwartz and S. Kraus. Bidding mechanisms for data allocation in multi-agent environments. In *Proc. Int. Workshop on Agent Theories, Architectures and Languages*, July 1997.

[32] M. Stonebraker, R. Devine, M. Kornacker, W. Litwin, A. Pfeffer, A. Sah, and C. Staelin. An economic paradigm for query processing and data migration in Mariposa. In *Proc. Int. Conf. on Parallel and Distributed Information Sys.*, Sep. 1994.

[33] M.P. Wellman, W.E. Walsh, P.R. Wurman, and J.K. MacKie-Mason. Auction protocols for decentralized scheduling. *Games and Economic Behavior*, 35:271–303, 2001.

[34] B. Wilcox-O'Hearn. Experiences deploying a large-scale emergent network. In *Proc. of the 1st International Workshop on Peer-to-Peer Systems*, March 2002.

[35] O. Wolfson, S. Jajodia, and Y. Huang. An adaptive data replication algorithm. *ACM TODS*, 2(2):255–314, June 1997.

Appendix A - Bid trading algorithm

Here we present one algorithm for conducting the bidding process. In CallAuction() the auctioning site contacts each remote site and asks for bids. Other algorithms are possible; for example, the auctioning site could broadcast the auction announcement and receive bids in parallel. PickWinner() determines which site wins the auction; the version presented here picks the lowest bidder. The GetBid() algorithm is run by a site that is bidding in an auction called by another site.

```
CallAuction(Collection C) {
  /* The size of the collection */
  S := C.size();

  /* The number of bids we receive */
  BidCount := 0;

  for each i := 0..n such that site s_i does not
    have a copy of C {
    D_i := Size of any deeds held by local
      site for site s_i's space;
    B_i := s_i.GetBid(S - D_i);
    if B_i != NULL then BidCount++;
    /* E.g., site i has refused to bid */
  }

  if (BidCount = 0) then return;
  /* No sites have bid */

  W := PickWinner(B_0..B_n);
  if (W = NULL) then return;
  /* All bids are too high.  */

  get a deed of size S - D_W from site W;
  give a deed of size B_W to site W;
}
```

```
PickWinner(Bids B_0..B_n) {
  L := LocalAvailableSpace();
  select lowest non-NULL bid B_i;
  if B_i > L then return NULL;
  return i;
}


GetBid(Size S) {
  L := LocalAvailableSpace();
  if S > L then return NULL;
  B := BidPolicy();
  return B;
}
```

In CallAuction(), the auctioning site only requests a bid for $S - D_i$ bytes from each site s_i. If the auction site already has a deed for D_i GB of space for a remote site ($D_i \geq 0$), then it only needs $S - D_i$ GB of space from that site. If site s_i wins the auction, then the auctioning site acquires a deed of size $S - D_i$ bytes, which, with the existing deed for D_i bytes, is enough to store the collection that the auctioning site wants to replicate.

The CallAuction() algorithm shows that the auctioning site calculates a value L, which is the local storage available for public use. The auctioning site uses a *space management policy* to determine L. The space management policy determine how much space to keep for itself, and how much to release for use by others. This released amount, L, is used as the locally "free" space, or space that can be traded away. Although it is possible to set L to be the total free local storage space, our previous work [8] (where the space management policy is called an "advertising policy") suggests that it is better to keep some of the local space in reserve for future use. (Although [8] focuses on fixed-price trading, it is reasonable to assume that the conclusion remains valid here.)

Our space management policy says that sites should "release" for public use space equal to $n \times a$, where n is the number of remote copies a site wishes to make and a is the amount of space used for archiving locally owned collections (e.g. measured in GB). For example, if a site wishes to make at least $G_M = 3$ copies, it needs to make at least 2 remote copies; thus, the space management policy is to release $2 \times a$ GB of space for public use, if possible. When a new collection of size s is deposited at the local site, this results in $2 \times s$ more public space being released at the local site.