Bidding for storage space in a peer-to-peer data preservation system

Brian F. Cooper and Hector Garcia-Molina

Department of Computer Science
Stanford University

{cooperb, 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 examine the best policies for determining when to call auctions and how much to bid, as well as the effects of "maverick" sites that attempt to subvert the bidding system. 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.*

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 copies and a failure at one site does not mean that information is lost forever.

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 [6]. 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 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 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 dif-

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

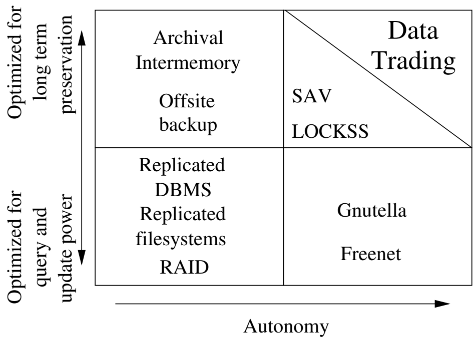

Figure 1. Classification of data management schemes.

ferent 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 [5] and LOCKSS [20], can be classified as community-based replication systems. This paper discusses how such systems can trade data to find the most reliable replication.

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 [17]: 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 [11]. These are some of the questions we address for the specific domain of reliable data replication in this paper.

Other distributed computing systems [22, 18, 10, 23] 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.

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.

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" for their own benefit. 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.

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

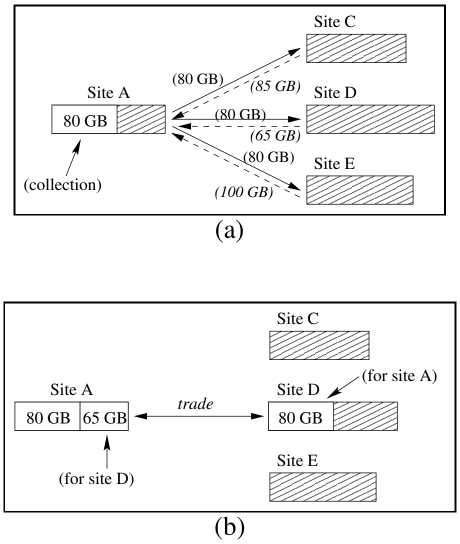

Figure 2. Bid trading example.

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 made G copies, where G is the site's *replication goal*. 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 individual decisions made at local 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 (Figure 2a). Each site responds with a bid B_i (italic values 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 (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 illustrated 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 $f_1, f_2, \ldots$, 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 to try to improve its own reliability even at the expense of the overall reliability. For example, one site may choose a different bidding function than that used by other sites in its class.
- *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. Since we are assuming trusted archival sites, we do not study in this paper mechanisms that enforce the selected policies or rules, or that detect violations.

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

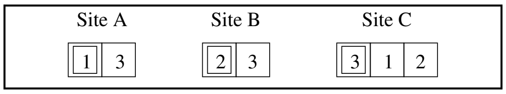

Figure 3. Reliability example.

```
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 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; }
```

Figure 4. Auction and bidding algorithms.

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 one 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, Figure 3 shows three sites, A, B and C, storing copies of collections 1, 2 and 3. The figure indicates (with a double box) that site A owns collection 1, site B owns collection 2, and site C owns collection 3. Each of the three sites (A, B and C) 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. Then the *site reliability* R_i, or probability that site s_i will fail each year, is 0.1. 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.

From the site reliabilities R_A, R_B..., and the assignment of copies to sites (shown in Figure 3), we can calculate the local data MTTF. First, we calculate the probability P_i that all copies of any collection owned by site s_i are lost in one year. The probability that collection 1 is lost is the probability of both sites A and C failing, or $0.1 \times 0.1 = 0.01$. Because this is the only collection owned by site A, $P_A = 0.01$. Next, we calculate the expected number of years M_i before any of site s_i's collections are lost. For site A, with $P_A = 0.01$, the MTTF M_A is 100 years. Similarly, the M_B of site B is 100 years, since collection 2 will be lost only if both site B and site C fail. However, M_C is 1000 years. For collection 3 to be lost, all three sites must fail, and this event has a probability of $P_C = 0.1^3 = 0.001$.

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 Figure 4. Other algorithms are possible; for example, the auctioning site could broadcast the auction announcement and receive bids in parallel.

The auction procedure finds all the remote sites that do not already have a copy of collection C, and solicits bids (via the GetBid() message) from these sites. Note that, for a site s_i, the auction site only needs $S - D_i$ GB of space, since it already holds deeds for D_i GB of s_i's space ($D_i \geq 0$). It is possible that some (or all) of these sites will not bid (submitting a bid of $NULL$), either because they do not have at least S space, or simply 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. Figure 4 shows a simple PickWinner() procedure that selects the site that submitted the lowest bid. In this scheme, the bidding site can bid less to have a better chance of winning the auction, but will get a smaller deed for space at the auctioning site in return. The auctioning site may extend PickWinner() to favor the most reliable bidding site, sites that have been dealt with before, or some combination of these and other factors. Here, we assume that the auctioning site simply picks the lowest bid. If the auctioning site does not have enough space to satisfy the lowest bid, there is no winner and the auction will terminate.

Figure 4 shows that the auctioning site calculates a value L, which is the local storage available for public use. The *space management policy* determines how L is calculated, for example to reserve some space for future use. For a detailed discussion of space management, see [6].

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, less or the same as S. At this point, 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 runs a local version of GetBid() to choose a bid 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 it, 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 *bidding policy*. Bidding policies are described in Section 3. (Note that if $S = B$, fixed-price trading occurs.) A basic version of GetBid() that uses a bidding policy, encapsulated in the function BidPolicy(), is shown in Figure 4.

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 interval of potential bids, and *P()*, the *policy function* that determines how bids vary along the interval I. $0 \leq P() \leq 1$. 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. The interval I reflects 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 I is $0.5 \times S...1.5 \times S$. 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.

More generally, a site can calculate its bid B as

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

This equation produces a *symmetric* bid policy: the midpoint of the interval I is at S. If the interval I is $S...S$, 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 [6], 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 those 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. However, it is possible that individual sites may pursue policies that benefit only themselves while causing a reduction in reliability for other sites. In the Maverick Site scenario, most sites use the policies that are best for their class, but one site deviates from these policies. Due to space limitations, maverick behaviors are examined in the extended version [7]. Experimental results for maverick sites are summarized in 6.4.

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_I	Ideal replication goal	6 copies

Table 1. Simulation variables.

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 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*. 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

*Our experiments indicate that although much larger networks are feasible, a larger number of sites is not necessary to achieve high reliability.

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). Sites call auctions as dictated by their auction call policies (see Section 3). Site failures are modeled with $R_i = 0.1$, as described in Section 2.1.

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 and the bid policy.

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 compared the effects of the auction calling policy. We ran five different experiments, one for each bid policy (including the FixedPrice policy). Our results (not shown) indicates that the CallForRare policy is always better, providing up to 850 ± 100 percent improvement in MTTF over the CallForAll policy (when $F = 5.8$). The results for other bid policies are similar: CallForRare is better than CallForAll regardless of which bid policy is used. These results suggest that it is detrimental to reliability if a site calls too many auctions. Although the CallForAll policy causes the site to actively try to replicate collections by calling auctions, sites call too many auctions too soon, using up their local storage, and too few copies are 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 happens with the CallForRare policy.

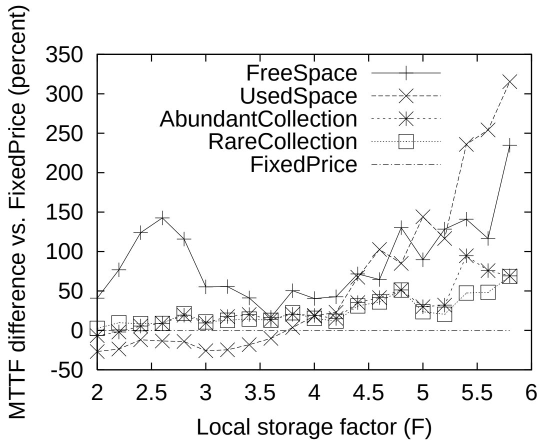

Figure 5. Bid policies in the Adaptive Bids scenario.

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. 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 [6].)

In our experiment, $I = 1$ and $P()$ was either FreeSpace, UsedSpace, AbundantCollection, or RareCollection for all sites; we also tested FixedPrice (e.g., $I = 0$). 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 5, 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 is best. For high capacity sites (with $F \geq 4.4$), either the UsedSpace policy or FreeSpace policy is 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 overlap and neither is 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 are roughly the same, since all are within the confidence interval of ± 50. For low capacity sites ($F < 3.2$), all policies are 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 win many auctions, or bid high and win big in a few auctions. Low capacity sites ($F < 2.6$), which often cannot bid at all, benefit from FreeSpace. When low capacity sites do bid, they tend to have little free space and thus bid aggressively, winning frequently. In the range $2.6 \le F < 4.4$, sites still cannot bid in very many auctions, but now tend to lose the ones they do bid in since they are bidding comparatively higher (since they have more free space). For $F \ge 4.4$, the "winning big" effect dominates, since high capacity sites can bid in many auctions and have a higher chance of participating in an auction they can 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 win auctions. Although these sites are not getting much remote storage *per auction* (because they bid low) they are winning many auctions, and get a large amount of remote space *in aggregate*. Low capacity sites win fewer auctions under UsedSpace because they are bidding higher. As noted above, low capacity sites only benefit by bidding aggressively, which they cannot do under UsedSpace.

This experiment suggests that 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 polices based on some partition of the sites. The results in Section 6.2.1 suggest that all sites benefit from the CallForRare policy, so we did not study the case where different classes used different auction policies. However, Figure 5 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 ($F \ge 4.4$), *mid capacity* sites ($3.2 \le F < 4.4$) and *low capacity* sites ($F < 3.2$).

We ran simulations in which all of the sites in one class used the same bid policy, while different classes may have used different policies. We can summarize the results as follows:

- The best class division is actually two classes, with low capacity $F \le 3.4$ and high capacity $F > 3.4$, rather than three classes. Recall that sites are trying to make at least $G_M = 3$ copies. This means a site with $F > 3.4$ has enough space to make 3 copies, and intuitively has a high storage capacity relative to the storage needed to make trades. A site with $F \le 3.4$ has trouble making 3 copies, and intuitively has low storage capacity relative to the needed storage.
- High capacity sites ($F > 3.4$) should use the UsedSpace policy with any $I > 0$. UsedSpace allows high capacity sites to bid low and win many auctions, so the sites can make as many copies as possible of their collections.
- Low capacity sites ($F \le 3.4$) should use the FreeSpace policy with $I = 2$. FreeSpace allows low capacity sites to bid low and win many of the auctions they participate in, so the sites can aggressively try to make at least 3 copies of their collections.
- Bid trading as a mechanism is useful, since it allows sites to improve their reliability over fixed-price trading.

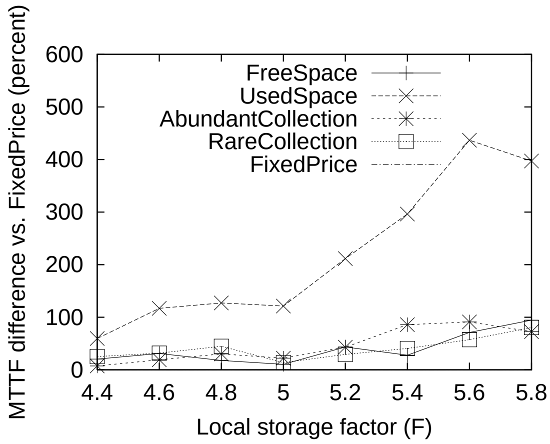

Figure 6. Best bid policy for high capacity sites.

In order to determine these results, we tested every combination of a possible bid policy for low, mid and high capacity sites; since there are five different policies and three storage classes there are 125 combinations. To start with, $I = 1$ in each case except FixedPrice. We analyzed the results by plotting the effect of bid policies on the reliability for a particular class for each combination of policies used by the other two classes.

For example, we plotted the effect on high capacity sites (where $F \ge 4.4$) of the bid policy used by those sites, in the scenario where mid capacity sites used the UsedSpace policy and low capacity sites used the AbundantCollection policy. The results are shown in Figure 6. As the figure shows, the UsedSpace policy is significantly better for high capacity sites than other policies. This general result, and the shape of the plotted curves, remains the same regardless of the bid policies used by mid and low capacity sites. Recall that under UsedSpace, sites bid low and win auctions when they have lots of free space. Even though high capacity sites make lots of trades, they tend to still have a lot of free space, and thus continue to win auctions and make copies of their collections. In other words, the high capacity sites have enough space so that they can afford to continually bid low. In this figure, the 95 percent confidence interval is ± 50

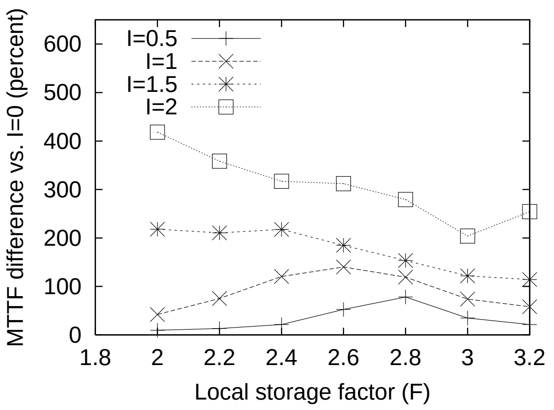

Figure 7. Best bid span for the FreeSpace policy and low capacity sites.

except in the range $F > 4.8$, where the confidence interval is ± 100. The dips and peaks in the UsedSpace curve are noise within these confidence intervals.

The results for other combinations of policies used by low and mid capacity sites are the same: UsedSpace is best for high capacity sites regardless of the policy used by other classes of sites.

For mid capacity sites (results not shown), UsedSpace is clearly the best policy for $F > 3.4$, but in the interval $3.2 \leq F \leq 3.4$ there is no clearly best policy. This suggests that $F > 3.4$ is a better definition of high capacity sites than $F > 4.4$. Further experiments confirm that two classes are appropriate: high capacity sites ($F > 3.4$) do best with UsedSpace and low capacity sites ($F \leq 3.4$) do best with FreeSpace, for the same reasons discussed in Section 6.2.2.

In order to examine the impact of 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 $I = 1$). The results (not shown) for high capacity sites indicates that the MTTF does not change significantly as I changes. Although sites achieve better reliability with $I > 0$, with up to 100 percent improvement in MTTF versus $I = 0$, the actual value of I does not matter.

We also ran an experiment where I varied between 0 and 2 for low capacity sites using FreeSpace, while high capacity sites use UsedSpace with $I = 1$. The results for are shown in Figure 7, which shows the percent difference in MTTF achieved by sites for each value of I versus a baseline of $I = 0$. As the figure shows, low capacity sites achieve the highest reliability with $I = 2$, with up to a 420 percent improvement over $I = 0$. By increasing the bid span, low capacity sites magnify the benefits of the free space policy: they win 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, in the hope of achieving benefit for themselves. This situation is the Maverick Site scenario. We have examined the behaviors outlined in Section 5 to see if such behaviors help the maverick site and hurt the other sites in the network. Due to space limitations, complete results are presented in the extended version [7]. In summary, our experiments show:

- A maverick high capacity site can sometimes benefit by always bidding high to try and extract a high price, but does not harm other sites doing so.
- A maverick high capacity site can also benefit by never calling auctions (instead only bidding in other auctions) and in doing so may harm other sites.
- Other behaviors do not benefit the maverick site, sometimes sharply reducing the maverick's reliability. Similarly, other behaviors do not hurt system reliability.

7. Related work

Previous investigators have studied distributed replication systems. Examples include traditional data management schemes, such as replicated DBMS's [3, 12], replicated filesystems [16, 13] and RAID disk arrays [19]. 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 [9, 21, 25]. Thus, traditional replicated databases tend to trade some reliability for increased performance [15]. Here, we are primarily concerned about preservation (given the constraint of preserving site autonomy).

Systems such as the Archival Intermemory [4] and OceanStore [14] are very good at preserving digital objects. These systems could use the techniques we describe here to determine where data will be replicated. Our work is also related to existing peer to peer trading systems such as FreeHaven [8] or Gnutella [1]. 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.

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 work is needed to apply theoretical mechanisms to real systems (as pointed out in [11]). 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 [24]. 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 [2]. 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.

Several systems have attempted to apply market-oriented programming, and specifically auction techniques, to resource allocation problems. Schwartz and Kraus [22] 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 [18], the Mariposa transaction processing system [23], and Ferguson, Nikolaou and Yemini's replicated data processing economy [10]. Our unique application, replication to achieve reliability, means that we can draw from this previous work but must also develop new techniques and policies.

8. Conclusion

We have described *bid trading*: a mechanism for allowing sites to conduct peer-to-peer data trading to achieve high reliability. Bid trading 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. Moreover, our experiments show that most malicious behaviors do not significantly harm the system. Our results suggest that bid trading is an effective, general model for peer-to-peer data trading and preservation.

References

[1] Gnutella. http://gnutella.wego.com, 2002.

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

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

[4] Y. Chen et al. A prototype implementation of archival intermemory. In *Proc. ACM Int'l Conf. on Digital Libraries*, 1999.

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

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

[7] B. F. Cooper and H. Garcia-Molina. Bidding for storage space in a peer-to-peer data preservation system (extended version). http://dbpubs.stanford.edu/pub/2002-22, 2002. Technical report.

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

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

[10] 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.

[11] D. Ferguson et al. Economic models for allocating resources in computer systems. *Market-Based Control: A Paradigm for Distributed Resource Allocation*, 1996.

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

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

[14] J. Kubiatowicz et al. OceanStore: An architecture for global-scale persistent storage. In *Proc. ASPLOS*, Nov. 2000.

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

[16] B. Liskov et al. Replication in the Harp file system. In *Proc. 13th SOSP*, Oct. 1991.

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

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

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

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

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

[22] 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.

[23] M. Stonebraker et al. An economic paradigm for query processing and data migration in Mariposa. In *Proc. Int. Conf. on Parallel and Distributed Information Sys.*, Sep. 1994.

[24] W. Walsh et al. Auction protocols for decentralized scheduling. In *Proc. ICDCS*, May 1998.

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