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

Brian F. Cooper and Hector Garcia-Molina

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.*

KEYWORDS: distributed storage management, data preservation, archiving, resource trading, auctions, bidding, data replication

TECHNICAL AREAS: fault-tolerance, reliability, agent systems, peer-to-peer networks

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 been developing a framework, called *data trading*, for replicating collections to achieve reliability, while allowing sites to make decisions about 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, 9]. 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 different decisions than a system that can centralize control in one place, or that places data close to users in order

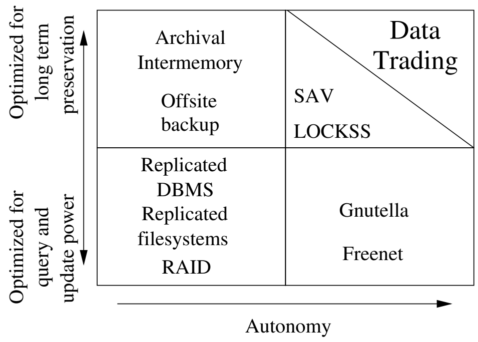

Figure 1. Classification of data management schemes.

to improve efficiency. Several systems, including SAV [7] and LOCKSS [24, 3], 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 [20]: 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 [26, 22, 13, 11] 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, rather than the resource itself, is the source of value. A local site has storage space of its own, but finds that storage space at a remote site is more valuable because it can be used to store a copy of a collection. This 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. In existing systems, sites must decide whether to keep a collection centralized, move the collection to a new location, fragment the collection, or make a copy of the collection, depending on the current access pattern. In our system, copies are made to ensure reliability, and the economic incentive system must be structured to maximize reliability, rather than access performance. Related work is discussed further in Section 5.

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 Section 3 we discuss polices for calling auctions and bidding, and ways in which maverick sites can deviate from "normal behaviors" for their own benefit. Section 4 presents the results of simulation experiments where we study the various policies and maverick behaviors. In Section 5 we examine related work, and in Section 6 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 contact another site (the "remote site") and propose a trade, 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 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 the trade: the amount of space that the local site must give to the remote site. In the simplest case, $S = B$, and the sites exchange equally sized deeds. We can call this scheme *fixed-price trading*. 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*. In this case, the local site calls an auction, announcing S, and remote sites $s_1, s_2 ... s_n$ respond with bids $B_1, B_2 ... B_n$. Bid B_i is the amount of storage the local site will have to reserve for the remote site in order for the remote site to devote S space to the local site. The local site can then choose the most attractive bid; this bid is usually the lowest B_i although other factors (such as how reliable the remote site is) may also affect the decision. The remote site chosen as the "winner" of the auction exchanges deeds with the local site.

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 80 GB to the remote sites (Figure 2a). Each site responds with a bid (Figure 2b); this bid is the amount of space site A will have to give to make a trade. Site A chooses the winner as site C, which submitted the lowest bid. Next a trade is conducted (Figure 2c), with sites A and C exchanging deeds. Now, site A can use its deed for site C 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.) Figure 3 shows this spectrum of freedom graphically.

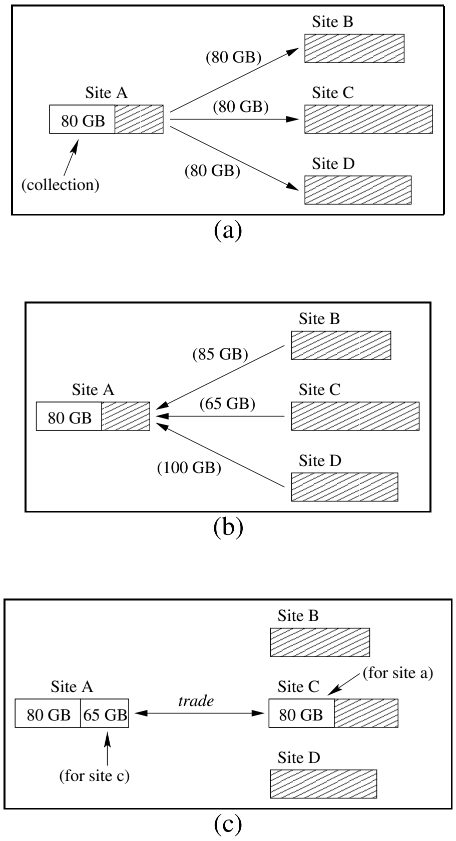

Figure 2. Bid trading example.

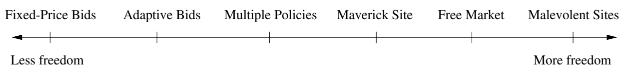

Figure 3. Spectrum of bidding freedom.

- *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 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

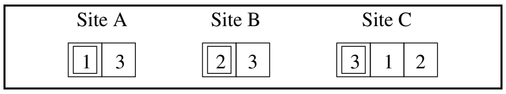

Figure 4. Reliability example.

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 spactrum, specifically the Fixed-Price, Adaptive, Multiple, and Maverick scenarios. Our main reason is that 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, we believe that the restrictive scenarios are realistic since archival sites will almost certainly want to trade with known entities they trust. Thus, it is reasonable to expect that sites would agree to use particular classes of bidding functions, ones that given them sufficient autonomy while still preserving the integrity of the overall system. 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. We can measure the reliability with which a collection is stored by calculating the probability that the object is not lost despite site failures. Therefore, we define the following reliability concepts:

Site reliability: the probability that a site will fail. By "fail" we mean a failure that results in data loss. A site can recover from such a failure by retrieving an error free copy of the lost data, usually from another site in the trading network. However, for some period after the failure, the data is locally irretrievable.

Local data mean time to failure (MTTF): the expected time that at least one copy of all of the local site's collections survives, despite site failures.

Figure 4 can be used to illustrate these reliability concepts. The figure 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 4), 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. For the example in Figure 4, collection 1 will be lost if both site A and site C fail, but will not be lost if either site survives. The probability of both sites failing is $0.1 \times 0.1 = 0.01$. 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. The probability that a collection is lost in the j^{th} year is $(1 - P_i)^{j-1} \times P_i$; that is, the probability that it was not lost in any of the previous $j - 1$ years but is lost in the j^{th} year. Then, we can calculate the mean time to failure for site s_i's collections as the expected number of years before its collections are lost, which is

$$M_i = \sum_{j=1}^{\infty}((1 - P_i)^{j-1} \times P_i \times j) = 1/P_i$$

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 $P_C = 0.1^3 = 0.001$.

Our goal here is to find which policies guiding the decision making of a local site maximize the local data MTTF for that site.

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. Then, the local site has taken on the role of the *auctioning site*. The decision of when to call an auction is determined by the *auction calling policy*. Auction calling policies are described in Section 3.1.1. An example of the steps that the auctioning site can take is shown in Figure 5. (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

```
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;
}
```

Figure 5. Auction algorithm.

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 5 shows a simple PickWinner() procedure that selects the site that submitted the lowest bid. Thus, 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. It is possible to extend PickWinner() to take more factors into account. For example, the auctioning site may prefer the most reliable bidding site, the bidding site with the most free space, or some combination of these and other factors. Here, we assume that the auctioning site simply picks the lowest bid. If the lowest bid is larger than the local free storage space, then the auctioning site cannot accept any bids, because it does not have enough space to give the bidding site. In this case, the winning bid is $NULL$, e.g. there is no winner and the auction will terminate.

Figure 5 shows that the auctioning site calculates a value L, which is the local storage available for public use. The

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

Figure 6. Bidding algorithm.

space management policy, which determines how L is calculated, is discussed in Section 2.3.

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 local site is now serving the role of the *bidding site*.) In the simplest case, the bidding site returns S, the auction amount. Then, the auctioning site and bidding site would exchange equally sized deeds, if the bidding site won the auction. However, 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 3.1.2.

A basic version of GetBid() that uses a bidding policy, encapsulated in the function BidPolicy(), is shown in Figure 6. This figure shows that the bidding site also calculates L, the amount of publicly available local space (from the space management policy). If the auction amount S is larger than L, the bidding site refuses to bid (returning a bid of $NULL$).

2.3. Space management policy

The auctioning site follows a *space management policy*[1] to 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 [9] suggests that it is better to keep some of the local space in reserve for future use. (Although [9] 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

[1] The space management policy is called an *advertising policy* in [9].

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.

3. Scenarios

The bidding mechanism provides a framework for data replication. However, sites must make two basic decisions: when to call an auction, and how much to bid in a particular auction. These decisions can be guided by an *auction calling policy* and a *bid policy*. In Section 2, we described the *Adaptive Bids* scenario, the *Multiple Policies* scenario, and the *Maverick Site* scenario. In this section, we examine the types of policies that may be adopted under each scenario.

3.1. Adaptive Bids scenario

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

3.1.1 Auction calling policies

The *auction calling policy* is a set of rules for automatically deciding when to call an auction and for what collection. The auction policy can either dictate that auctions are called periodically, or that they are called in response to some event. For example, a site may call an auction every night, or may call auctions when a new collection has been deposited locally. Here, we assume that sites call auctions when they need to make copies of their collections, and when they believe that there is a good chance that an auction will result in a trade. Thus, if a site calls an auction and no remote sites bid (e.g. because the remote sites do not have enough storage space), then it does not make sense to call the auction again unless the state at the remote sites change (e.g. at least one site gets more space). Therefore we focus on event-based auction calling policies. We are not concerned here with the mechanisms that sites use to detect events. Instead, we can assume that the implementation of the system allows events to be detected.

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 an auction to replicate each collection, and can do so 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.

3.1.2 Bid policies

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.
- *P()*: the *policy function* that determine 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*.

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. Then, $P() \propto K$, where K is the fraction of local storage space still free.

A special case of this family of I-P policies is where the interval I is $S ... S$. In this case, the bidding site always bids S (and $P()$ is immaterial). This is the *fixed-price* policy. If all sites use the fixed-price policy, then there is no bidding, and the result is fixed-price trading (as described in Section 2).

Here, we examine bid policies with different values of I and $P()$. For $P()$, we examine a set of policies based on two factors: the amount of available local storage space at the bidding site, and the rareness of the bidding site's collections.

In order to construct a bid, the bidding site must determine how valuable it feels its local space is in relation to the remote site's space. In other words, it must determine an *exchange rate* between the two space resources. We can represent this exchange rate E as a ratio of the value of the bidding site's space to value of the auctioning site's space. For example, if $E = 2$, then the bidding site feels that every unit of its own space is twice as valuable as every unit of the remote site's space. In this case, if the auctioning site asks for 3 GB of space at the bidding site, then the bidding site should ask for 6 GB in return.

The site's bid B can thus be calculated as

$$B = E \times S$$

The exchange rate E can vary from auction to auction depending on the current situation of the bidding site. For example, the bidding policy may adjust E upward as local space gets used up to indicate that space is more valuable as it becomes scarcer. The bidding policy determines the value of E for each auction, and thus determines how a site bids.

I defines the maximum and minimum bids allowed by the policy, while $P()$ determines the bid for a particular auction. In this paper, our goal is to study how I and $P()$ impact the reliability a site is able to achieve, so that we can define bid policies that produce the highest reliability. For simplicity, here we assume that bid policies are *symmetric*: the interval I straddles the value 1. In this case, E varies in the range $(1 - I/2)...(1 + I/2)$. Then, we can calculate E as

$$E = I \times P() + (1 - I/2)$$

and B as

$$B = S \times (I \times P() + (1 - I/2)) \quad (1)$$

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 [8, 9], 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 the results of Section 4, we compare the reliability achievable under bid trading to those achievable under fixed-price bidding.

3.2. 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. Therefore, it may be useful to partition the sites into distinct classes, and allow each class to use a different policy. This is the Multiple Policies scenario. For example, we may create a class of sites that have a large amount of storage space, and another class of sites that have less storage space. Then, 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. The Multiple Policies scenario is less restrictive than the Adaptive Bids scenario, where all sites must use the same policy regardless of needs or resources.

For the Multiple Policies scenario, we can study the same alternatives outlined in Section 3.1. 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.

3.3. 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.

Although there are a large number of ways in which a maverick site may attempt to subvert the trading mechanism for its own benefit, we can only examine a few here. Specifically, we will examine some ways that sites may use the ability to call auctions and choose their own bids to "unfairly" benefit themselves. In Section 4.4 we examine the effect (if any) these behaviors have on the reliability of sites in the trading network. Recall that we are focusing here on behaviors that still fit within the rules of the protocol, and are not examining behaviors which are malicious and violate the basic trading framework.

3.3.1 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 try to do this in an attempt to reserve as much space as possible 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 hoard all of the space at remote sites.

NeverCall: a site never calls auctions. A local site may try to do this so that it could 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.

3.3.2 Maverick bidding behaviors

Under normal bid policies, sites sometimes bid high ($E > 1$) and sometimes bid low ($E < 1$). However, maverick sites may decide to pattern their bidding in order to take advantage of other sites, rather than calculating bids based on normal bid policies. Here, we study the *BidHigh*, *BidLow* and *NeverBid* behaviors.

BidHigh: a site consistently bids high; $E > 1$ 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. This behavior allows a site to extract more resources from the system than it is contributing. If every site bid high always, then what a site gained when it was a bidder it would lose as an auctioner, and no site would gain benefit. If a BidHigh site accrues advantage, it is because it is the only site consistently bidding high.

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
$Ctot$	Total data at a site	$Ctot_{MIN}$ to $Ctot_{MAX}$
$Ctot_{MIN}$, $Ctot_{MAX}$	Min/max value of $Ctot$	$Ctot_{MIN} = 200$ GB, $Ctot_{MAX} = 10,000$ GB
G_M	Minimum replication goal	3 copies
G_I	Ideal replication goal	6 copies

Table 1. Simulation variables.

BidLow: a site consistently bids low, e.g. $E < 1$ always. The benefit of BidLow is that a site wins more auctions, and thus reserves more space at remote sites for its own use. 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.

4. 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.

4.1. 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 initial base 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 as well as a number of collections "owned" by the site. The number of sites S is specified as an input to the simulation. Our experiments represent 1200 total scenarios, 200 for each S in the range $10...15$. The number of collections assigned to a site is randomly chosen between $CperS_{MIN}$ and $CperS_{MAX}$. All of the collections in the system are ordered randomly and are deposited at their assigned site in this 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. The collections assigned to a site 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 is the *total data size* $Ctot$ of that site, and 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 our experiments, we wanted to study the effect of bid policies on sites with a large amount of space (relative to their collection size) and on sites that had comparatively less space relative to data size. Therefore, in each scenario, some sites may have a large F (e.g. 6) while others may have a small F (e.g. 2). Although a particular site is assigned a quantity of storage space, it does not release all of this space immediately for public use. Instead, the site follows the space management policy described in Section 2.3.

Sites call auctions in response to events indicating that the global state has changed (see Section 3.1.1). In our simulator, the basic events occur when a user deposits a new collection at a site. The site receiving the new collection calls n auctions to replicate the new collection; the value of n is dictated by the site's auction policy. At the same time, other sites also call auctions as dictated by their auction call policies. The auctions called by different sites are randomly interleaved to model a series of auctions being called concurrently by multiple sites. Depositing collections are the main state-changing events because the space management policy dictates that new space is released at a site after it gets a new collection. Thus, a deposit of a new collection is a signal that there is now more space in the system, and previously impossible trades may now be feasible.

As described in Section 2.1 we model site failures by specifying a value R_i: the probability that site s_i will fail. In the present work, for all sites $R_i = 0.1$. (For experiments where the site reliability differs, but specifically in the context of fixed-price trading, see [8].)

In the following sections, we examine the improvement or detriment due to using one policy versus another. For example, imagine a site achieves a MTTF of 100 years using policy X, and a MTTF of 300 years using policy Y. Then, 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).

4.2. 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.

4.2.1 Auction policies

The auction policy dictates when a site will call an auction, and for which collection. As described in Section $3.1.1$, we examined the CallForAll and CallForRare policies. 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 Fixed-Price policy). A sample result for the auction policy, in a situation where sites used the UsedSpace policy, is shown in Figure 7. This figure shows the average increase or decrease in reliability experienced by sites when the CallForRare policy is used versus a baseline of when the CallForAll policy is used. The CallForRare policy provides up to 850 percent improvement in MTTF over the CallForAll policy (when $F = 5.8$). The 95 percent confidence interval for this figure is ± 50 percentage points, except for $F > 4.4$, where the interval expands to as much as ± 100.

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, the end result is that 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, and bidding in auctions called

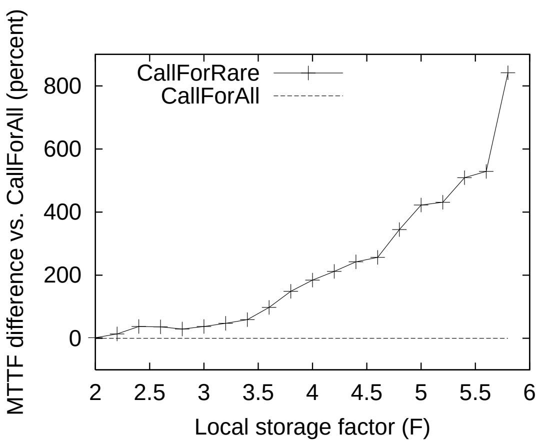

Figure 7. Auction calling policies under the Adaptive Bids scenario.

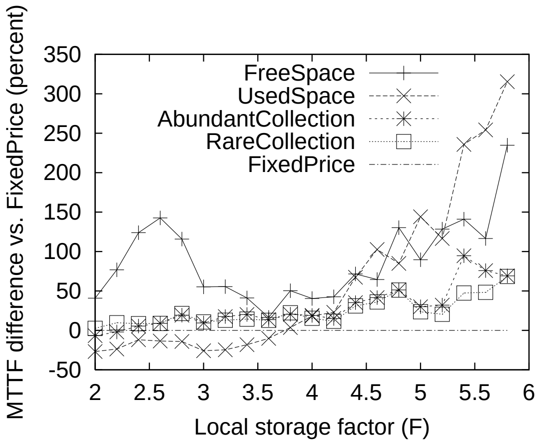

Figure 8. Bid policies in the Adaptive Bids scenario.

by other sites. The first copies of a collection can be made by calling auctions, in order to ensure that the collection is replicated at least a few times. Then, the collection can be replicated more times when the site bids in and wins auctions. This is what happens with the CallForRare policy.

4.2.2 Bid policies

Next, we examined different bid policies. The bid policies described in Section 3.1.2 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 [9].)

In our experiment, $I = 1$ and $P()$ was either FreeSpace, UsedSpace, AbundantCollection, or RareCollection for all sites; we also tested FixedPrice (e.g., $I = 0$). The FixedPrice policy represents a trading network that does not use bidding, and 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 8, which shows the percent MTTF change for each bid policy versus a baseline of the FixedPrice policy. 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.

Low capacity sites tend to have little free space, and thus these sites bid low (and win auctions) under the FreeSpace policy. Very low capacity sites ($F \leq 2.6$), can rarely bid in auctions (since they usually do not have enough space to store the auctioning site's collection) and must aggressively try to win all the auctions they do participate in. This means that FreeSpace benefits low capacity sites. As F increases, sites tend to have more local storage available, which means that sites with higher F bid less aggressively, winning fewer auctions. This causes the downturn seen in the FreeSpace curve of Figure 8 for $F > 2.6$. However, the FreeSpace curve begins to rise again for $F > 4.4$, e.g. for high capacity sites. These sites, with a large amount of free space, bid high and win few auctions. However, when they do win an auction, they "win big," getting a large amount of remote space while giving little away in return. Therefore, there are two competing effects: bidding low and winning many auctions, or bidding high and winning big in a few auctions. Low capacity sites, which often cannot bid at all, benefit from FreeSpace because when they do bid, they bid aggressively. In the range $2.6 \leq F < 4.4$, sites still cannot bid in very many auctions, but now tend to lose the ones they do bid in. For $F \geq 4.4$, the "winning big" effect dominates, since high capacity sites can bid in many auctions and thus can afford to wait until they can win an auction 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.

4.3. Multiple Policies scenario

In the Multiple Policies scenario different sites use different polices based on some partition of the sites. The results in Figure 7 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 8 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 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 $F \leq 3.4$ 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 $F \leq 3.4$ 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 ($F \leq 3.4$) 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.

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

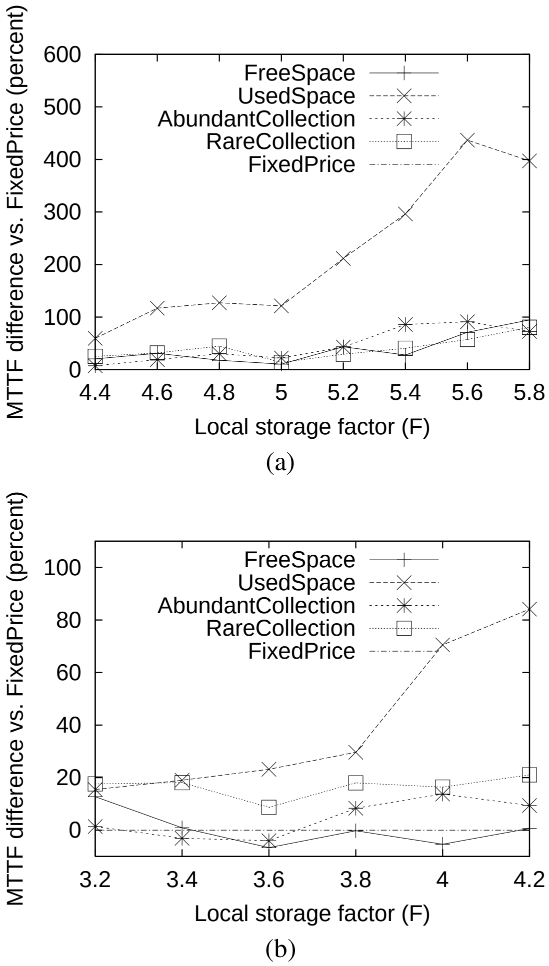

Figure 9. Best bid policy for (a) high capacity and (b) mid capacity sites.

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 $F \geq 4.4$) 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 9a. 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 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.

Sample results for mid capacity sites (in the case where high capacity uses UsedSpace and low capacity uses FreeSpace) are shown in Figure 9b. (The 95 percent confidence interval in this figure is ± 30.) As this figure shows, 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. The same result is observed regardless of the low capacity bid policy used. (We restricted our examination to the cases where high capacity sites use UsedSpace, since it is clearly the best policy for those sites.) This suggests that $F > 3.4$ is a better definition of high capacity sites than $F > 4.4$. Moreover, further experiments (results not shown) suggest that the interval $3.2 \leq F \leq 3.4$ is best considered part of the low capacity class; all sites in the range $F \leq 3.4$ do best with the FreeSpace policy when high capacity sites ($F > 3.4$) use UsedSpace. In other words, high capacity sites do best with UsedSpace and low capacity sites do best with FreeSpace, for the same reasons discussed in Section 4.2.2.

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. Increasing I increases the maximum bid that the high capacity site makes. While this means that the site receives more remote storage when it wins an auction, it also means the site wins fewer auctions because it is more likely that some other site is bidding less. The simulation results indicate that these effects cancel out.

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 10, 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.

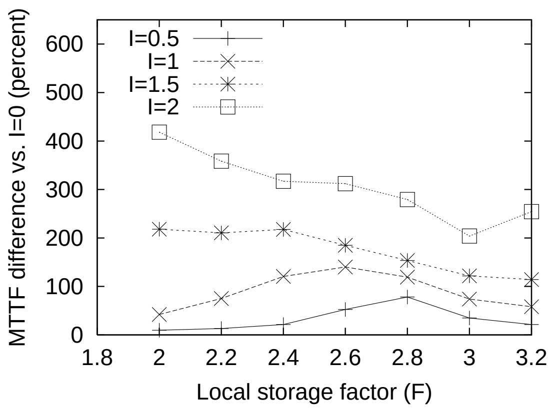

Figure 10. Best bid span for the FreeSpace policy and low capacity sites.

4.4. 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. Here, we consider whether the behaviors outlined in Section 3.3 can be used to benefit an individual site. Specifically, we are interested in two questions:

- Can a site accrue benefit by behaving in a manner different from the rest of its class?
- Does the differing behavior reduce the reliability achieved by the sites that are following their class's policy?

In other words, it is not useful to a site to act differently from its class if it achieves no benefit. At the same time, it may not detrimental for normal sites if one site's behavior deviates. We study these questions in this section.

We have implemented maverick behaviors as described in Section 3.3. With the BidHigh behavior, the maverick site uses $E = 1.5$ always, and with the BidLow behavior, the maverick site uses $E = 0.5$ always. With the AlwaysCall behavior, the maverick site continually calls 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 calls auctions and never bids 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 how much a site can benefit itself or damage the system by acting differently.

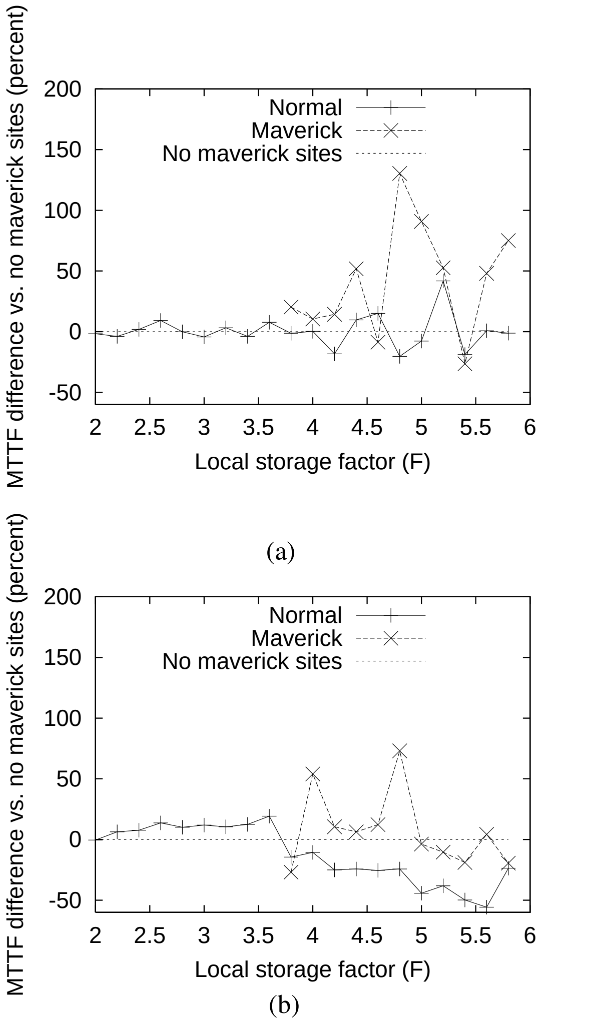

Figure 11. Maverick behaviors: (a) BidHigh and (b) NeverCall.

The results were:

- A maverick high capacity site can sometimes benefit from the BidHigh behavior, but does not harm other sites doing so.
- A maverick high capacity site can also benefit from the NeverCall policy, and in doing so may harm other sites.

We first examined the situation where a maverick high capacity site deviates 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 11a. This 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 varies 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 does not deviate. Moreover, the variance in our measurements is very high: the 95 percent confidence interval for the "Maverick" curve is ± 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 experience 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 is frequently the lone bidder in an auction, and thus acquires a large amount of remote space at little cost to itself. In other sessions, there are usually more bidders in an auction, and thus the maverick site wins few auctions, makes fewer trades and experiences a loss in reliability. The end result is that the BidHigh behavior is risky; sometimes it pays off and sometimes not.

However, Figure 11a also indicates that non-maverick sites do 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 is able to extract a high price in an auction, other sites are 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 11b shows the results from another experiment, where one high capacity site pursues the NeverCall behavior. As with the BidHigh behavior, the maverick site sometimes does well (achieving up to a 75 percent increase in MTTF) and sometimes does poorly (achieving up to a 25 percent decrease in MTTF). Once again, the variance is 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 uses the UsedSpace policy, often bidding low and winning auctions. When the site's storage space begins to fill up, the site starts losing auctions, because it is bidding higher. Normally in this situation, a site still trades by calling auctions, but must often pay a high price in these trades (since the remote site sets the price.) However, a maverick site refuses to call auctions, instead bidding (and bidding high). If the maverick site is the only bidder, then it gets a large amount of remote space and makes several copies of its collectons. If there are other bidders, the maverick site loses auctions and makes no trades. Thus, as with the BidHigh behavior, sometimes NeverCall benefits a maverick site and sometimes hurts the site. This produces 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 is either winning auctions at high prices and reserving much of the space in the system for itself, or losing auctions and therefore not giving away its own space. In either case, some sites that may otherwise use this space cannot, resulting in less reliability for those sites. This is seen most clearly in Figure 11b 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 and refuse to trade with it altogether, encouraging the maverick site to pursue normal behavior.

Our experiments have also shown that other maverick behaviors are not effective, resulting in either no benefit or sharply reduced reliability for the maverick site. BidLow is not effective because although a site wins many auctions, it always does so by giving away much of its own space and getting little in return. NeverBid is also ineffective because the local site is at the mercy of bids cast by other sites. In other words, every auction the site participates in results in a trade (because the site is the auctioner) but many of these trades come at a loss for the local site if other sites are bidding high. AlwaysCall is not effective for two reasons. First, 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 uses up all of its local space without necessarily replicating many of its collections. Second, AlwaysCall is like the NeverBid behavior in that most trades are a result of the maverick site calling an auction, and then potentially paying a high price in the trade.

In no case does a maverick behavior benefit a low capacity site. 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 in an auction is key to benefiting from the BidHigh or NeverCall behaviors.

4.5. Number of sites

All of the results reported here are for relatively small peer-to-peer networks of 10 to 15 sites. A small network is appropriate for our problem domain, where we assume a small federation of libraries and archives cooperating to provide preservation. A library is unlikely to entrust its collections to thousands of Gnutella-like clients running on unknown, unreliable home computers. Instead, the library will choose a set of remote sites that are relatively well trusted, and conduct bid trading among these sites.

In previous work [8], we have examined the impact of the site count on reliability for the Fixed Price scenario, and found that a relatively small network of about 5-7 sites is in fact the most reliable. In the context of this paper, we have conducted experiments under the Adaptive Bids, Multiple Policies and Maverick Site scenarios, and found that once again 5-7 sites is the optimal network size. These results suggest that a larger network of sites can achieve high reliability by forming trading groups of 5-7 sites.

5. Related work

Previous investigators have studied distributed replication systems. Examples include traditional data management schemes, such as replicated DBMS's [5, 15], replicated filesystems [19] and RAID disk arrays [23]. 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 [10, 25, 28]. Thus, traditional replicated databases tend to trade some reliability for increased performance [18]. Here, we are primarily concerned about preservation (given the constraint of preserving site autonomy).

Similarly, replicated filesystem schemes such as Coda [16] or Andrew [21] use caching to improve availability. Andrew and Coda treat replicates as cached copies that are created on demand and ejected from the cache when necessary. Data trading places data in response to reliability needs, and we assume that a site accepting data is making a long term commitment to provide access.

Systems such as the Archival Intermemory [14, 6] and OceanStore [17] are very good at preserving digital objects. High replication is achieved at the cost of site autonomy, as sites do not have control over where their collections are replicated or which remotely-owned collections they store.

Our work is also related to existing peer to peer trading systems such as Freenet [1] or Gnutella [2]. Such systems are focused on finding resources within a dynamic, ever-changing collection, and not on reliability, and less popular or infrequently accessed items can be deleted. Thus, systems like Gnutella provide searching but do not guarantee preservation. A searching and resource discovery mechanism could be built on top of our data trading system; however, our primary focus is surviving failures over the long term.

Auction theory, in both economics and computer science, has been extensively developed. 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 [12]). 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 [27]. Other investigators have looked at "efficient clearing", or the best way to assign resources to bidders so as to maximize utility across the system. In this scenario, methods such as integer programming [4] can be used to solve the auction, but this assumes all resources and bids are known at the same time. In our system, resources and bids appear over time as new collections are created and new storage is added, and archives, which must 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 [26] 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 [22], the Mariposa transaction processing system [11], and Ferguson, Nikolaou and Yemini's replicated data processing economy [13]. Our unique application, replication to achieve reliability, means that we can draw from this previous work but must also develop new techniques and policies.

6. Conclusion

We have described *bid trading*: a mechanism for allowing sites to conduct peer-to-peer data trading to achieve high reliability. Collections are replicated when two sites agree to trade space, such that each site can store data using the other site's storage space. 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. This results in a situation where a site calls an auction when it wants to trade. Other sites submit bids, and the auctioning site chooses the lowest bid.

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. Although the CallForRare policy is good for all sites, there is no one bid policy that is universally most reliable. Bid trading with the UsedSpace policy provides the highest reliability for sites with a lot of storage capacity. Sites with less storage capacity should use the FreeSpace policy instead. We have also shown that if some sites deviate from the recommended policy for their class, they may benefit themselves slightly but only in some cases damage the reliability of other sites. Our results suggest that bid trading is an effective, general model for peer-to-peer data trading and preservation.

References

[1] The Freenet Project. http://freenet.sourceforge.net/, 2001.

[2] Gnutella. http://gnutella.wego.com, 2001.

[3] LOCKSS status. http://lockss.stanford.edu/-projectstatus.htm, 2001.

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

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

[6] Y. Chen, J. Edler, A. V. 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.

[7] B. 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.

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

[9] B. Cooper and H. Garcia-Molina. Peer-to-peer data trading to preserve information. *ACM Transactions on Information Systems*, to appear.

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

[11] M. S. 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.

[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] A. Goldberg and P. Yianilos. Towards an archival intermemory. In *Advances in Digital Libraries*, 1998.

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

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

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

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

[19] 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.

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

[21] J. H. Morris, M. Satyanarayanan, M. H. Conner, J. H. Howard, D. S. H. Rosenthal, and F. D. Smith. Andrew: A distributed personal computing environment. *CACM*, 29(3):184–201, March 1986.

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

[23] 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.

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

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

[26] 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.

[27] W. Walsh, M. Wellman, P. Wurman, and J. MacKie-Mason. Auction protocols for decentralized scheduling. In *Proc. ICDCS*, May 1998.

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