Peer to peer data trading to preserve information

Brian Cooper and Hector Garcia-Molina

{cooperb,hector}@db.stanford.edu
Department of Computer Science
Stanford University

Abstract

Data archiving systems rely on replication to preserve information. In this paper, we discuss how a network of autonomous archiving sites can trade data to achieve the most reliable replication. A series of binary trades between sites produces a peer to peer archiving network. We examine two trading algorithms, one based on trading collections (even if they are different sizes) and another based on trading equal sized blocks of space (which can then store collections.) We introduce the concept of *deeds*, which track the sites that own space at other sites. We then discuss policies for tuning these algorithms to provide the highest reliability, for example by changing the order in which sites are contacted and offered trades. Finally, we present simulation results that reveal which policies are most reliable.

1 Introduction

Digital information is best preserved by replicating it at archives run by *autonomous organizations.* For instance, if a single publisher stores a collection of journals (even with backup tapes), there is a danger that the publisher might go out of business and the information may become inaccessible. It is much safer for several publishers or libraries to jointly archive the journals. Similarly, it is best if the copies are managed by different software, so that a common error, say in one brand of file system, does not corrupt all copies. This is the way today's libraries work with paper copies, and several projects [17, 8, 27, 14] have been extending this preservation approach to digital data.

To achieve data preservation in a network of cooperating archives, we propose the concept of *data trading*: sites replicate their data by contacting other sites and proposing trades. Thus, technical reports owned by site A may be replicated to site B. In return, a copy of a collection of software documentation owned by site B is stored at site A. In such a network, each site makes local decisions about which other sites to contact and offer trades as well as whether to accept trades offered by other sites. The result is a global peer to peer archiving network, built up from a series of locally agreed upon binary trading links.

To illustrate such a data trading mechanism, consider the example shown in Figure 1. Figure 1(a) shows sites A and B, each of which have two gigabytes of space, and site C, which has three gigabytes of space. (A gigabyte is represented in the figure as a box.) Site A owns a collection of data labeled "1," site B owns collection "2," and site C owns collection "3." (A collection is an application unit, e.g., a set of technical reports, or a set of census files.) Each collection requires one gigabyte of space. Sites A and B can trade their collections, resulting in the configuration of Figure 1(b). Collections 1 and 2 are now stored more reliably, because if one site goes out of business, goes on strike or burns down, another copy is available. However, now site C cannot trade with either A or B since neither site has free space for collection 3. Thus, collection 3 is not stored reliably.

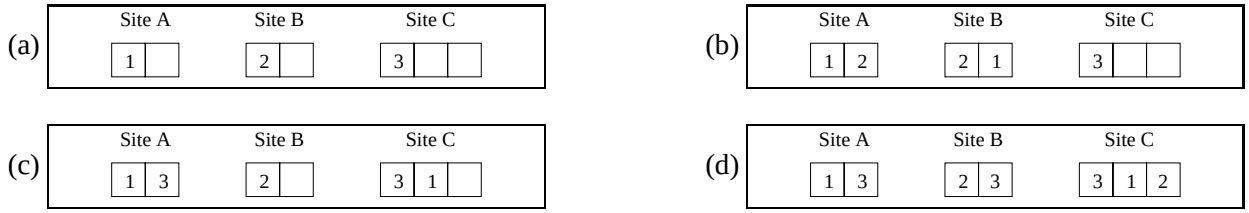


Figure 1: Data trading example.

A different trading order can result in a more desirable scenario. For example, say that from the initial configuration, site A first contacts C and offers a trade. The result is shown in Figure 1(c). Now there is still enough space to make another trade, this time between sites B and C. The resulting situation, in Figure 1(d), has all three collections reliably stored with two copies. A trading scheme should be effective enough so that sites can make local decisions about which sites to trade with, while still allowing other sites to replicate their collections. At the same time, trading must be flexible enough to deal with the appearance of new sites, new collections, and even new free storage added at an existing site.

There are a number of interesting variations for trading which lead to different reliabilities and implementation costs. In particular, sites can trade collections or blocks of space. With collection trading, complete collections are exchanged, as in our previous example. With blocks of storage, the "buckets" that hold data are traded, but they do not have to be filled in immediately. We use the term *deed* to refer to an empty block; it represents the right to use a certain amount of space at a given site. Deeds can be combined to obtain larger blocks of space, and they can be traded with third parties.

Collection trading is simple to implement and manage; however, it may not be fair because the traded collections may be of different sizes. When trading blocks of space, a site can ask for a large enough block to contain a collection it wants to replicate, and then give an equal sized block in return. Trading blocks of space requires more bookkeeping to record blocks that have been traded but not yet used.

In this paper we study these two trading strategies, and the design decisions that must be made to implement them. For example, in what order does a site look for trading partners? How much free space does a site "advertise" when making trades? When and how are deeds traded? We also study the impact of key parameters, such as the number of participating sites and the amount of free space available, on the achievable reliability. We evaluate designs and scenarios with a detailed trading simulator. The simulator lets us mimic large numbers of trading sessions to find expected reliabilities.

Our work draws upon concepts developed in related systems. Figure 2 shows a schematic classification of data management schemes, including our work and some other sample systems. (Section 6 discusses more related work.) 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).

For example, the lower left box of Figure 2 contains traditional data management schemes, such as replicated DBMS's [4, 18], replicated filesystems [12] and RAID disk arrays [25]. 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. We do not believe that autonomous sites will, or should, surrender such decision

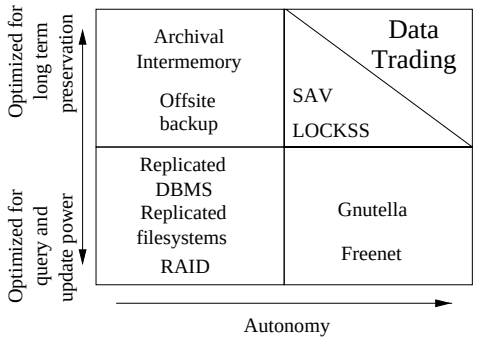


Figure 2: Classification of data management schemes.

making. Also, we are concerned with "fairness:" a site that provides a relatively high amount of free space should expect a correspondingly higher level of reliability. Fairness and autonomy are much less important for disk arrays and file systems, since typically all disks are owned by the same organization. Another difference is that traditional solutions are concerned with load distribution, query time and update performance, as well as reliability. Here, we are primarily concerned about preservation (given the constraint of preserving site autonomy). Thus we are willing to trade some access performance for increased reliability and site independence.

The lower right box of Figure 2 shows 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. While popular items may become widely replicated, less popular or frequently accessed items are 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.

Systems such as the Archival Intermemory [17, 15] (in the upper left box of Figure 2) 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. For example, under our mechanism, the MIT library can tell its funders that they are hosting ACM journals and Berkeley reports, as opposed to saying that they are hosting some ever-changing set of files. Moreover, the MIT library can refuse to store information that it deems improper or illegal.

Another possibility is for archives to contract for an offsite backup, in which another organization agrees to store a copy of the data, usually for a fee. Such a scheme provides reliability, but the backup lacks autonomy, because the offsite backup must store the data given to it by the archive site. Offsite backups may be quite expensive, and offer no guarantee of preservation if the archive goes bankrupt and no longer pays its bills. Under a data trading scheme, an archive achieves the high reliability of having multiple offsite backups, while only having to buy extra local data storage. This is especially important for libraries and other organizations that are on a limited budget, and require a data preservation scheme that can utilize their existing resources. Libraries that already provide digital access to their collections will not have to buy additional infrastructure but can augment the systems they already have with the data trading mechanisms we describe here.

Another difference between data trading and many of the existing systems mentioned above is that it is often very desirable to keep collections contiguous and immovable. It is much easier to manage and index a collection if each of its copies is at a single site, rather than fragmented and scattered around the network. Similarly,

management is simpler if a collection moves rarely. We expect collections to be large, so the cost of movement across geographically distributed sites would be high. Therefore, in this paper we only study algorithms that keep collections together and that do not move the collections once they have been copied. This is the reason that in the example in Figure 1, we did not consider "undoing" site A's mistake (trading first with site B) by moving previously traded collections. Many of the systems in Figure 2 do allow collections or files to be fragmented and moved. For example, in a RAID system, files are often fragmented to improve read/write times.

The upper right box of Figure 2 shows systems such as SAV [8] and LOCKSS [27], which are good at both preserving site autonomy and ensuring long term data archiving by utilizing a community-based approach to preservation. All community members will benefit if the system is able to preserve information, including their own. Thus, if one site experiences a failure (temporary or permanent), then the remaining sites are motivated to preserve the data owned by the failed site. Moreover, organizations such as libraries have a commitment to preserving information, even if it is "owned" by other sites. This is an additional incentive to protect the data. Motivations such as these have led over 40 libraries and 30 publishers to participate in LOCKSS [3]. A community preservation approach could utilize the data trading mechanism we describe here to achieve a highly reliable data distribution. Currently these systems either require administrators to manually configure data replication (in the case of SAV) or distribute data based on accesses while potentially failing to preserve rarer materials or collections of specialized interest (such as with LOCKSS).

In summary, in this paper we describe how data trading among autonomous sites can be used to achieve high reliability. Specifically, we make the following contributions:

- We define two basic algorithms, COLLECTION_TRADING and DEED_TRADING, for automatically conducting trades between autonomous data archiving sites. We also propose the concept of *deeds* as a mechanism to facilitate trading blocks of space.
- We identify ways to tune the operation of both the COLLECTION_TRADING and DEED_TRADING algorithms; for example, by changing the order in which sites should be contacted and offered trades.
- We present results of simulation experiments that show the impact on reliability of each alternative policy. This allows us to choose the most reliable policies and thus effectively tune the trading algorithm.

Our focus here is on protecting the "bits" of the data. Other aspects of preservation, such as metadata, translating formats or protecting intellectual property rights can be dealt with by layers operating on top of the mechanism we describe here. This paper is organized as follows. In Section 2, we discuss our model, and in Section 3 we present the COLLECTION_TRADING and DEED_TRADING algorithms, as well as the tunable policies for each. Section 4 describes our simulator, and Section 5 presents results and discusses tradeoffs. In Section 6 we review related work.

2 Data preservation overview

Redundancy is the primary mechanism for achieving preservation. Making several copies of important data is critical to ensure that the data can survive multiple failures. Our model of a preservation system based on redundancy contains the following components:

- A network of *archiving sites*. Sites are autonomous units, managed by different organizations, and thus are not under any centralized control. New sites can be created at any time.

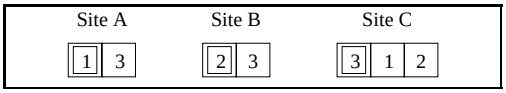


Figure 3: Reliability example.

- Each site has a quantity of *archival storage*: storage that is devoted to storing archived data, including copies of data that originated at other sites. The site provides *online access* to this archival storage. New archival storage may be added to a site at any time.
- The basic unit of archived data is the *data collection*: a set of related data, for example a group of files, a database table, a set of documents, etc. Collections may contain any number of data items, and different collections may be different sizes. Collections may also grow over time, as new data is added.
- Archiving sites have *clients*: users or applications that connect to the site to utilize archiving services. A client archives data by depositing a collection at a site; the site is then said to *own* the collection and takes primary responsibility for preserving the collection. A client can deposit a new collection at any time. Clients can also read archived data at the owning site or at a site storing a copy of the collection.

We also need a method for determining *reliability*. The goal of an archiving system is to reliably protect data, despite the potential for site failures. Thus, we use the following concepts:

- *site reliability*: The probability that a site will not fail.
- *local data reliability*: The probability that the collections owned by a particular site will not be lost.
- *global data reliability*: The probability that no collection owned by any site will be lost.

Figure 3 can be used to illustrate these reliability concepts. Each of the three sites (A, B and C) could fail independently. If we assume (for example) a 0.1 probability of site failure, then the *site reliability*, or probability of no failure, is 0.9 for each site. We can then calculate the *local reliability* for each site. In Figure 3, the collections owned by a particular site are shown double boxed; thus site A owns collection 1, and so on. 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 $0.1 \times 0.1 = 0.01$, so the probability of one or both sites surviving is $1 - 0.01 = 0.99$. This is the probability that collection 1 will not be lost. Because this is the only collection owned by site A, the *local reliability* of site A is 0.99. Similarly, the local reliability of site B is 0.99, since collection 2 will be lost only if both site B and site C fail. However, the local reliability of site C is 0.999. For collection 3 to be lost, all three sites must fail, and this event has a probability of $0.1^3 = 0.001$. As a result, the probability that at least one site survives is $1 - 0.001 = 0.999$.

We calculate *global reliability* by determining the probability that any collection is lost, regardless of which site owns it. In the example of Figure 3, all collections will survive if site C survives. This event has a probability of 0.9. If site C fails, an event with probability 0.1, then the collections will survive only if both site A and site B survive. The probability of both sites avoiding failure is $0.9 \times 0.9 = 0.81$. The global probability is then

$$P(\text{C lives}) \times P(\text{All data survives given C lives}) + P(\text{C fails}) \times P(\text{All data survives given C fails}) = 0.9 \times 1 + 0.1 \times 0.81 = 0.981$$

I. The local site L repeats the following until the replication goal <G> for collection C_L is met, or until all sites have been contacted and offered trades:

1. Select a remote site R according to the trading strategy <S>.
2. Contact R, and request that R advertise its free space. The amount of free space A_R advertised by R is determined by R's advertising policy $<A>_R$.
3. If R cannot store C_L (because $A_R < size(C_L)$), then the trade cannot be completed. L returns to step I.
4. L advertises its own free space A_L to R according to L's advertising policy $<A>_L$.
5. R selects a collection C_R that it owns, where $A_L \geq size(C_R)$, according to the collection copy policy <C>.
   - If no collection exists that meets the condition $A_L \geq size(C_R)$, the trade cannot be completed. L returns to step I.
6. The trade is executed, with L storing a copy of C_R and R storing a copy of C_L.

II. If the goal <G> is not met for collection C_L, L may decide to try again in the future, according to the retry policy <R>.

Figure 4: Algorithm COLLECTION_TRADING

Thus, even though each local site reliability is ≥ 0.99, the global reliability is only 0.981. Each site can justify its expenditure of resources by noting that it has achieved very high reliability for its collections, e.g. ≥ 0.99. The designer of a trading mechanism would consider the global reliability of 0.981 to evaluate design choices.

In this paper, we assume for simplicity that all site reliabilities are equal. We have conducted experiments where site reliabilities differ, and the trends are the same as those reported here. We have also generalized our algorithms to take into account the possibly different reliabilities of sites when making trades; due to space limitations, those results are reported elsewhere [9] and summarized in Section 5.6. We also assume that there is some mechanism to protect the security of the data, for example using encryption to ensure the privacy of sensitive material that is replicated to other sites.

3 Data trading

In our data trading scheme, a site that wishes to make copies of a data collection contacts another site and proposes a trade. We refer to the *local site* as the site that actively makes a connection to another site and proposes a trade. The site contacted by the local site is called the *remote site*; the remote site can accept or refuse the trade proposal. If the trade succeeds, each site devotes a portion of its archival storage to storing the other site's data.

One possibility is that sites trade *collections*. For example, site A may agree to store a copy of site B's collection of technical reports, and in return, site B stores a copy of site A's image collection. Under this approach, the traded collections may be different sizes.

We assume here that collections have a set size, although different collections may have different sizes. The scenario where collections grow can be handled by assigning a fixed size to the collection that is larger than its actual size; this provides extra space for growth. If this space is exhausted, then a new collection is created to contain the new data, and a new round of trading is conducted to replicate this new collection. Finally, we assume in this paper that collections, once traded, are not moved (see Section 1).

The first algorithm, COLLECTION_TRADING, is shown in Figure 4. It is conducted separately by each local site wishing to make a trade. The algorithm starts when a client contacts the local site L and deposits a

collection C_L to be archived. COLLECTION_TRADING can be tuned using several policies, each of which cause the algorithm to behave differently: <G>, <S>, <A>, <C>, and <R>.

<G>: The *replication goal* of the local site. In this paper we assume that the replication goal is that each site wants to make N copies of a collection. The generalization of our algorithm to other replication goals is straightforward.

<S>: The *trading strategy* of the local site. The trading strategy determines the order in which remote sites are contacted and offered trades. We describe the strategies in Section 3.1.

<A>: The storage space *advertising policy* followed by a site. A site may advertise all of its free storage space, or may advertise only a fraction of the space and reserve the rest for future trades. This issue is discussed in more detail in Section 3.2.

<C>: The *collection copy* policy. When a remote site is offered a trade, it must select one of its collections to copy to the local site. For example, it may decide to replicate only collections that have not met the replication goal <G>. In this paper, we assume that the remote site always selects the rarest collections to copy first, even if these collections have enough copies to meet the goal <G>.

<R>: The *retry* policy followed by a site. A site may use *active retries*, in which it initiates COLLECTION_TRADING again at some future time for collections that do not meet the replication goal <G>. The site would maintain a list of collections that had not met the goal, and retry these collections in response to a trigger (e.g., every week). In contrast, a site may limit itself to *passive retries*: after running COLLECTION_TRADING once for a collection, a site waits to be contacted by other sites to make further copies.

COLLECTION_TRADING can be illustrated by the following example. At first, there is one archiving site, site A, with 12 Gb of space. A client contacts site A, and deposits collection 1, which requires 1 Gb of space. Site A stores collection 1, and now has 11 Gb of free space. However, in the absence of any other sites, site A cannot replicate the collection. Now, a second archiving site is created, site B, with 6 Gb of space. A client contacts site B and deposits collection 2, which requires 2 Gb of space. Site B now has 4 Gb free, and contacts site A to propose a trade. Site A agrees to store collection 2, and in return, site B will store a copy of collection 1. Now, there are two copies of both collections 1 and 2, as each is stored at both sites 1 and 2. This process continues with more collections and more sites being created at any time.

The alternative to collection trading is to trade blocks of space. These blocks of space will be used to store collections, although the sites may decide not to exchange data immediately. For example, a local site may acquire a block of space at a remote site; in trade, the local site reserves a block of its own space for use by the remote site. The local site may use a portion of the block at the remote site to make a copy of an existing collection, but save the rest for future collections. The local site may also decide not to use any of the space it has just acquired, but to save all of it for the future. This flexibility results in added complexity compared to collection trading, and a bookkeeping mechanism is required to manage this complexity.

The mechanism we propose is called *deeds*, an analogy to property deeds. A deed is a right to use a block of space at a particular site. This deed may be used, in whole or in part, at any time by the site holding the deed. If site A holds a deed for 500 megabytes of site B's space, site A may exercise a portion of the deed now to store a copy of 100 megabytes of images, and reserve the remaining 400 megabytes. Deeds may also be transferred, in whole or in part, to other sites. Thus, site A may *split* its deed for site B into two deeds, one of size 200 megabytes and another of size 300 megabytes. Site A can then transfer the 200 megabyte deed to site C, which can then use the deed to store its own collections.

The algorithm for trading blocks of space utilizes deeds; it is called DEED_TRADING and is shown in Figure 5. This algorithm is similar in some respects to COLLECTION_TRADING, and in particular can be tuned with policies <G>, <S>, <A> and <R>. However, there are additional parameters for DEED_TRADING: <D>, <U>, <T>, and <S_T>.

<D>: This is the *deed size policy*, and represents the amount of space each site will request from the other. The simplest policy is that L selects a deed size (D_L) equal to the size of collection C_L minus the size of any deeds L already has for R, and R picks an identical size (D_R). This is the policy we assume in this paper. However, the algorithm could be generalized to the case where $D_L \neq D_R$. This might occur for example if one site was more reliable than the other, and thus the less reliable site may have to give more space to compensate.

<U>: This is the *deed use policy* which determines when a site will use a deed it has acquired. A local site running DEED_TRADING will use the deeds it acquires to immediately replicate the collection C_L. However, the remote site R will acquire a deed that it may use immediately or at some future time. Clearly, if R has rare collections it should use the deed to propagate them. However, if R does not have rare collections, it must decide what to do. If R is aggressive, it might use the deed immediately to replicate collections, even if they are not rare. In contrast, R might decide to save the deed for use with new collections that it acquires in the future.

<T>: This is the *deed transfer policy*. This policy either allows deeds to be transferred among sites, or requires that the original holder of a deed be the one that uses it. If <T> allows deeds to be transferred, a site running DEED_TRADING may decide to propose trades to acquire deeds held by other sites at any time. However, a site should only acquire deeds held by other sites if it facilitates data replication; see Section 3.3.

<S_T>: The is the *third party trading strategy*. If the remote site does not have enough undeeded free space, the local site may decide to collect deeds held by other sites. The local site chooses an order to contact *third party* sites which hold deeds for the remote site. That order is determined by the third party trading strategy. Here, for simplicity, we assume that the local site uses the same strategy used to select remote sites, that is, <S_T>=<S>. It is straightforward to extend our mechanism to allow sites to choose a different strategy for contacting third party sites; in other words <S_T>$\neq$<S>. Several strategies are described in Section 3.1.

We can illustrate DEED_TRADING by extending the example above. Site A (12 Gb of storage) is given collection 1 (requiring one gigabyte of space) and stores it, but cannot make trades until new sites are created. Site B is created with 6 Gb of space, and is given collection 2 (requiring two gigabytes). Site B stores collection 2, and then contacts site A, proposing a trade for deeds worth two gigabytes. Site A agrees to this trade, and reserves two gigabytes of its space for site B, while site B likewise reserves two gigabytes of its space for site A. Site B uses its deed to store collection 2 at site A; the deed is now used up. Site A uses its deed to store collection 1 at site B, but afterwards still has a deed worth one gigabyte for site B. Now, a client contacts site A and deposits collection 3, of size three gigabytes. Site A, which has 9 Gb of space remaining, uses three of those gigabytes to store collection 3. Site A contacts site B and proposes a trade, but only requests a two gigabyte deed, since A still holds a deed for site B worth one gigabyte. Site B, which has two gigabytes of free space remaining, agrees. Site B's space is now used up, and site B holds a two gigabyte deed for site A. This process continues, although B will not be able to store new collections or make new trades unless it acquires more local space.

Clearly, DEED_TRADING is more complex than COLLECTION_TRADING. Moreover, deeds may introduce more fragmentation, as supposedly "free" space is rendered unusable by virtue of being reserved under a deed. On the other hand, DEED_TRADING seems more "fair," since equal amounts of space are always traded, and judicious use of deed transfer could reduce the fragmentation problem. We have evaluated both algorithms

I. The local site L repeats the following until the replication goal <G> for collection C_L is met, or until all sites have been contacted and offered trades:

1. Select a proposed deed size D_L, according to L's deed size policy <D>$_L$.
2. Select a remote site R according to the trading strategy <S>.
3. If L has a deed for R then:
   (a) If the existing deed is $\geq size(C_L)$ then:
      i. Use the existing deed to store a copy of C_L. Return to step I.
   (b) If the existing deed is $\leq size(C_L)$ then:
      i. Set $D_L = D_L - size$(existing deed).
4. Contact R, and request that R advertise its free space. The amount of free space A_R advertised by R is determined by R's advertising policy <A>$_R$.
5. If $A_R < D_L$ then:
   (a) If the transfer policy <T> allows deeds to be transferred from one site to another:
      i. L may contact a *third-party site* T (that is, a site other than R) and trade some of L's storage space for T's deed for R. L may have to conduct several such trades to acquire enough deeds; if so, it contacts them according to the third party trading strategy <S$_T$>.
      ii. If the total of the deeds acquired through third-party trades, plus the deed L already had for R, plus the free space A_R, is enough to store collection C_L, the trade can be completed. Otherwise, the trade cannot be completed and L returns to step I.
      iii. L adjusts D_L downward such that $A_R \geq D_L$, subject to the constraint that the total of the deeds held by L for R plus D_L is still greater than or equal to $size(C_L)$.
   (b) If <T> does not allow transfers, the trade cannot be completed and L returns to step I.
6. L advertises its own free space A_L to R according to L's advertising policy <A>$_L$.
7. R selects a proposed deed size D_R, according to R's deed size policy <D>$_R$.
8. If $A_L < D_R$ then the trade cannot be completed. L returns to step I.
9. The trade is executed, with L acquiring a deed of size D_L for R's space, and R acquiring a deed of size D_R for L's space.
10. L uses its deeds for site R to store a copy of C_L.

II. If the goal <G> is not met for collection C_L, L may decide to try again in the future, according to the retry policy <R>.

III. At any time site may use a deed that it posesses to replicate one of its collections. A decision to use a deed is based on the deed use policy <U>.

Figure 5: Algorithm DEED_TRADING

Strategy	Description
Random	Trade with sites in random order
First fit	Trade with sites according to a globally defined order
Neighbors	Trade with sites according to a locally defined order
Clustering	Trade preferentially with sites that we have traded with before
Best deed	Trade first with sites for which we have the smallest deed
Worst deed	Trade first with sites for which we have the largest deed
Best fit	Trade first with sites that have the least free space
Worst fit	Trade first with sites that have the most free space
Neediest	Trade first with sites that have the rarest collections

Table 1: Trading strategies (<S>).

in terms of reliability. We have also evaluated various choices for the tuning parameters (<S>, <A> and so on). Section 4 describes our evaluation methodology, and Section 5 discusses our results.

3.1 <S>: Trading strategies

The order in which a local site contacts remote sites can have a significant impact on the reliability achieved. We have designed several strategies for selecting trading partners; they are listed in Table 1. Although some of the strategies are drawn from previous work in memory allocation (e.g. best fit), we are applying them in a new distributed, negotiation based context. Moreover, others, such as the neediest strategy, have no analog in memory allocation. In this section, we describe these strategies. In Section 5.3, we evaluate them in terms of reliability.

The strategies we have examined fall into two categories. The *random*, *first fit*, *neighbors*, *clustering*, *best deed* and *worst deed* strategies allow a site to make decisions using only local information. Local information includes a record of sites previously traded with, and a knowledge of deeds held for other sites. In contrast, the *best fit*, *worst fit*, and *neediest* strategies require up-to-date information about the global state of the system, such as how much free storage each site has and how many copies have been made of each collection. Of course, all strategies require information about which sites have been created. We assume that a newly created site announces its existence in some way, and that knowledge about which sites exist is stored locally at each site.

For the strategies that require global information, there must be some way of acquiring an up-to-date picture of the global state. One way of doing this is to contact all sites and request the needed information. Another possibility is to maintain a central information repository that is updated by a site when its state changes. Then, any site could contact the repository and obtain a snapshot of the global state. A third possibility is that sites could broadcast state information to other sites, for example whenever there is a local change. We are not concerned here with the actual mechanism for obtaining global information. Instead, we assume that the global state can be examined by a site wishing to make a trade, and our concern is how best to use this information. The strategies which require global state may run counter to the distributed, autonomous nature of the system, especially if they require a centralized information repository. We have examined them here to understand whether trading can be truly distributed or if some global state maintenance mechanism must be used to achieve the best reliability.

A site using the *random* strategy randomly selects sites to trade with. The order of sites contacted varies from collection to collection.

In the *first fit* strategy, sites are always tried in the same order, and this order is global. For example, sites could be contacted in order by IP address or domain name.

The *neighbors* strategy is like first fit except that each site contacts other sites according to a local ordering. Each site can choose its own local order of remote sites to contact, and follows this order for each trade. The first fit strategy will cause a few sites to be tried first for every trade, and those sites will likely be overloaded. The neighbors strategy attempts to avoid this phenomenon.

The placement of copies can be important in addition to the number of copies. The *clustering* strategy attempts to take advantage of this fact. For example, imagine that there are four sites, sites A-D, which are each given one collection, collections 1-4. There are many ways to distribute copies of these collections, including:

I.

Site A	Site B	Site C	Site D
1 2	1 2	3 4	3 4

II.

Site A	Site B	Site C	Site D
1 2	2 3	3 4	4 1

Configuration I is more reliable with a global reliability of 0.98, as compared to 0.96 for configuration II. This difference is due to the fact that after a site failure in configuration II, a failure at either of the "neighboring" sites will cause data loss, while in configuration I, a second failure must come at one specific site to cause loss. This argument suggests that clustering copies of collections is important. For example, if a cluster of collections 1 and 2 occurs at one site, it is beneficial to make a cluster of 1 and 2 at other sites. Under *clustering*, a site trades preferentially with the same sites it has concluded trades with before, in an attempt to keep its collections clustered. If there are ties, e.g. two sites could be chosen to achieve clustering, the local site can use any strategy to break the ties. Moreover, if no clusters exist yet, some strategy must be used to select new partners to trade with. We assume here for simplicity that the *random* strategy is used to break ties and to select new partners.

In the *best deed* and *worst deed* strategies, a site examines the deeds it is already holding before deciding which site to trade with. These strategies are only applicable if DEED_TRADING is being used. Under *best deed*, a site uses the smallest deed first, while under *worst deed* a site uses the largest deed first. Note that in both cases, the held deed may be large enough for the collection being replicated, and the deed may be then used without requring any actual trading. However, if the deed is smaller than the collection size, the local site will have to trade with the remote site to make up the difference. We assume here that for both best deed and worst deed, the random strategy is used if there are no deeds, and that ties are broken randomly.

The *best fit* and *worst fit* strategies require global state information about the free space available at remote sites. The *best fit* strategy attempts to find the site whose free space best fits the collection to be replicated. Specifically, if a site wishes to replicate a collection of size N units, then all sites with at least N units of free space are selected and then sorted in order of increasing free space. In contrast, a site using the *worst fit* strategies will contact sites with the most free space first, and contact other sites in order of decreasing free space. We assume here that for both best fit and worst fit, ties are broken using the random strategy. Also, if DEED_TRADING is being used and deed transfer (<T>) is allowed, then a local site includes the size of unused deeds in the best fit or worst fit calculations. This space is effectively free since the local site can trade with the deed holders to acquire the deeds if necessary.

Finally, in the *neediest* strategy, a site contacts the neediest site first. The neediest site is defined as the site with the rarest collection, e.g. the collection with the fewest copies. The local site must use global state information to determine which sites are needier than others. Again, we assume that ties are broken randomly.

3.2 Space advertising

A site must decide how much of its local space should be offered for use by other sites. In the ideal situation, a site always has enough space to store collections deposited by local clients, while simultaneously offering the optimum amount of space to remote sites. A simple solution would be to advertise the total amount of free local space. However, if this space becomes used up, the site will no longer be able to store collections deposited by local clients, and thus will fail to provide archiving to its users.

Therefore, a better solution would be to partition the site's total space (called T) into a *local* portion L_{total} (for use by local clients) and a *public* portion P_{total} (to store collections owned by remote sites). When trading, the local site advertises all or part of its public space. A good partition may be hard to find. If L_{total} is too small, the local portion can become used up, preventing local clients from depositing collections. If L_{total} is too large, the public portion can become used up, preventing the site from making trades. Ideally, L_{total} is no larger than the total size of collections that will be deposited locally, although this value may be hard to predict.

Even if the local space is "ideally" partitioned, the local site will be unable to make trades if the public portion becomes filled up. Thus, a local site should slowly release the public space for use by others. Instead of advertising all of the available public portion all of the time, the local site would advertise only a fraction initially, and increase the amount advertised as more trades were made.

There are two different ways to advertise the public portion so that space is released over time. One way, which we call the *space-fractional* policy, advertises a fixed fraction of the amount of free public space. Then, if the unused part of the public space is P_{free}, sites would advertise:

$$x \times P_{free} \text{ where } (0 \leq x \leq 1)$$

If a site chooses $x < 1$, then a fraction of the public space is released while the rest is kept in reserve for the future. If a site chooses $x = 1$, then it always advertises all of its free public space.

The second way to advertise public space is called the *data-proportional* policy. In this policy, a site releases public space to reflect the amount of locally owned data. The motivation behind this policy is that by releasing new public space to correspond with newly deposited collections, a site ensures that it has some public space to trade away to replicate those collections. If the portion of L_{total} and P_{total} that has been used is L_{used} and P_{used} respectively, then a site advertises:

$$MIN(y \times L_{used} - P_{used}, P_{free})$$

A site chooses a factor y, e.g. $y = 2$, and releases y times the amount of locally owned data. As more data is archived locally, the amount of released space increases. Some of this released space is used to store collections owned by remote sites, so the value advertised is $y \times L_{used} - P_{used}$. The site can never advertise more public space than it has; hence, the advertised amount should not exceed P_{free}.

3.3 Deed transfer

The DEED_TRADING algorithm allows sites great flexibility in using the deeds they have acquired. This flexibility includes the ability to transfer a deed to another site that needs it. For example, site A may wish to trade with site B, but find that most of site B's space is reserved under deeds held by other sites. Site A can then contact these sites and try to acquire deeds for site B, and site A can then use these deeds to store its collection.

Step 5 of Figure 5 describes how a site would acquire deeds from third parties. There may be other reasons why a site transfers deeds; for example, a site may wish to trade away deeds it holds instead of giving away precious local space. We do not consider these situations in this paper.

Note that deed transfer is only useful if the local site can acquire enough deeds to meet its needs. Trading is a parallel, distributed process, with each local site executing DEED_TRADING concurrently and independently. Thus, even if enough deeds exist to meet a local site's needs, the site must be able to acquire them all by making separate, binary trades with each of the third party sites. If another site in a similar situation acquires some of the deeds before the local site can, then the local site may be unable to accumulate enough deeds to replicate its collection at the remote site. There are a number of mechanisms that could be used to deal with this situation. The local site could contact each third party site and reserve the deeds before trading for any of them. Then, if the local site is unable to reserve enough deeds, it can release all of the reservations. This prevents a site from acquiring some deeds, and finding them to be useless since the rest are unexpectedly taken by another site. Another possibility is to have a central deed registry, where a local site can both find out which deeds are available and atomically acquire all of them[1]. A third possibility is to simply make as many trades as possible, and hope that enough deeds can be acquired to meet the local site's needs. We are not concerned here with the specific mechanism used to reserve deeds, but instead with the reliability that could be achieved if deeds were effectively acquired by sites that need them.

4 Trading simulation

In order to evaluate the different variations of data trading, we have developed a simulation system. This system generates a trading scenario, and then conducts a series of simulated trades using a given set of trading policies. The global and local reliabilities of the resulting collection distribution are then calculated. Table 2 lists the key variables in the simulation and the initial base values we used; these variables are described below.

The simulation works as follows. 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. For most of our experiments we use $S = 15$, but in Section 5.5 we consider other values and their impact on system reliability. The number of collections assigned to a site is randomly chosen between $CperS_{MIN}$ and $CperS_{MAX}$, and 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* $Ctotal$ of that site, and ranges from $Ctotal_{MIN}$ to $Ctotal_{MAX}$. The values we chose for $CperS_{MIN}$, $CperS_{MAX}$, $Csize_{MIN}$, $Csize_{MAX}$ and $Ctotal$ represent a highly diverse trading network with small and large collections and sites with small or large amounts of data. Thus, it is not the absolute values but instead the range of values that are important. The archival storage space assigned to the site is some multiple F_L of $Ctotal$ at the site[2]. The ratio of the global storage space to the global total amount of data is F_G. In some experiments, we set $F_L \approx F_G$ such that every site has roughly the same ratio of space to data. In other experiments, we vary F_L widely while keeping F_G fixed; e.g. $F_G = 4$ and $2 \leq F_L \leq 6$. Unless explicitly indicated otherwise, experiments reflect the case where $F_G \approx F_L$.

[1] As noted above, the idea of a central registry runs counter to the distributed, autonomous nature of sites. We mention it here for completeness, but a more distributed solution may be desired.

[2] The amount of archival storage may be influenced by other factors besides data size, such as cost.

Variable	*Description*	*Base values*
S	The number of sites	2 to 15
F_G	The global free space factor	2 to 7
F_L	The local free space factor	2 to 7
P	Site reliability	0.9
$CperS_{MIN}$, $CperS_{MAX}$	Min and max collections per site	$CperS_{MIN} = 4$, $CperS_{MAX} = 10$
$Csize_{MIN}$, $Csize_{MAX}$	Min and max collection size	$Csize_{MIN} = 50$ Gb, $Csize_{MAX} = 1000$ Gb
$Ctotal_{MIN}$, $Ctotal_{MAX}$	Min and max amount of data at a site	$Ctotal_{MIN} = 200$ Gb, $Ctotal_{MAX} = 10,000$ Gb
<G>	Replication goal	3 copies
<S>	Trading strategy	9 strategies tried
<A>	Advertising policy	fixed or increasing
<R>	Retry policy	active and passive, or passive only
<D>	Deed size policy	$size(C_L)$ (C_L =the local site's collection)
<U>	Deed use policy	aggressive or non-aggressive
<T>	Deed transfer policy	transfer allowed or not allowed

Table 2: Simulation variables.

The scenario also contains a random order in which collections are created and archived. The simulation considers each collection in this order, and the "owning" site replicates the collection. A site is considered "born" when the first of its collections is archived, and thus all of the sites begin trading at different times. A site does not have advance knowledge about the creation of other sites or collections. Our results represent 100 different scenarios for each experiment. For each scenario, the simulator runs simulated trading sessions for each policy setting. Here, all sites use the same policies. In ongoing work we are examining the impact of different sites using different policies, including so-called "rogue sites" that pursue local benefit to the detriment of the global system. However, space limitations prevent us from examining such issues here.

We model site failures by specifying a value P: the probability that a site will not fail. This value reflects not only the reliability of the hardware which stores data, but also other factors such as bankruptcy, viruses, hackers, users who accidentally delete data, and so on. History has shown that any of these factors can cause data loss, and may be more likely than a simple disk failure. In this paper, $P = 0.9$. We have conducted experiments with scenarios that may have different values of P for different sites as well as strategies that may take these differing values into account. Due to space limiations, we do not consider such situations here; results are reported in [9].

In this paper we assume that the replication goal <G> of a site is to make three copies of its collections. As the example of Figure 3 shows, a collection that has three copies is stored very reliably with only a 0.001 probability of data loss.

5 Results

We have evaluated the COLLECTION_TRADING and DEED_TRADING algorithms, as well as the policies used to tune the algorithms, using our simulator. Our evaluation is primarily concerned with reliability. We report results for both global and local reliability. Since we are concerned here with designing a trading mechanism to provide the best reliability for all sites, we tend to focus on global reliability. Moreover, for experiments where local reliability is not reported, the trends observed were not different between local and global reliability although

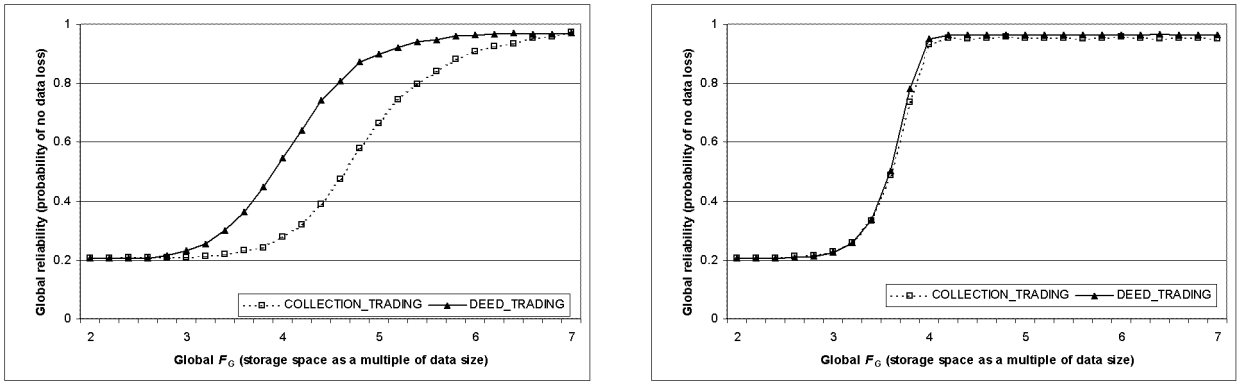


Figure 6: COLLECTION_TRADING versus DEED_TRADING algorithms.

the magnitude of the reliability values differed. We report more results involving local reliability in [10, 9]. Other factors, such as the complexity of implementing the trading algorithms, are not considered here.

In the following sections, we evaluate each design choice presented in Section 3, and then summarize the recommended policies for each algorithm. Due to space restrictions, we do not give discussion for the retry policy <R> or the deed use policy <U>. Results for these policies are summarized in Section 5.6 and full discussion is available in the extended version of this paper [10].

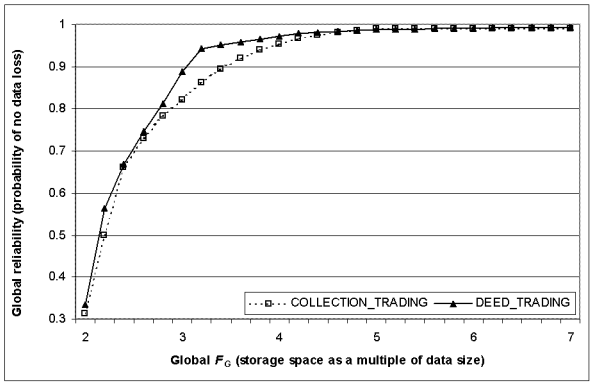
5.1 COLLECTION_TRADING versus DEED_TRADING

We start by studying our basic trading algorithms, COLLECTION_TRADING and DEED_TRADING, in particular a scenario with 15 sites:

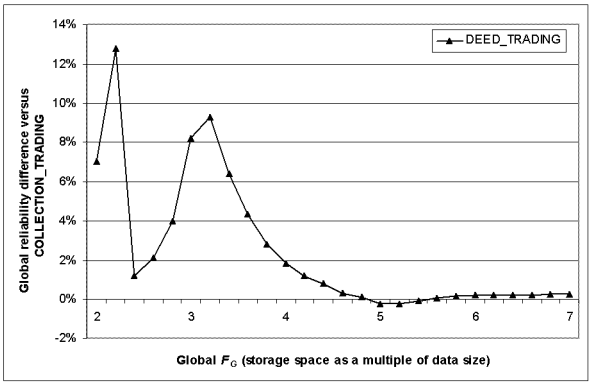
- <S>: Each site uses the *random* strategy to select trading partners.
- <A>: Each site uses the space-fractional advertising policy, with $x = 1$.
- <R>: A site restricts itself to passive retries only.
- <U>: A site uses its deeds to replicate as many collections as possible (*aggressive* deed use).
- <T>: Sites can transfer deeds if needed.

Figure 6(a) shows the global reliability as a function of the space factor F_G. As expected, the system achieves higher reliability as the amount of storage space increases, and sites are able to make more copies of their collections. Although both algorithms eventually achieve very high global reliability (≈ 0.99), DEED_TRADING is able to achieve high reliability (≈ 0.9) at $F_G = 5$, while COLLECTION_TRADING requires more space, as much as $F_G = 6$, to achieve equivalent reliability. Thus, DEED_TRADING appears to be the more desirable algorithm because it provides higher global reliability using less space.

Note that the difference in reliability shown in Figure 6(a) is quite significant; even smaller reliability differences can have a large impact. Consider the difference between 0.99 and 0.98 global reliability, a difference of one percent. We can interpret these values as the probability that data will not be lost over the course of some interval, for example, one year. Then, we can calculate the mean time to failure (MTTF) of the system



(a)



(b)

Figure 7: Potential maximum reliability of COLLECTION_TRADING and DEED_TRADING algorithms.

by computing the expected number of years before data is lost. A global reliability increase from 0.98 to 0.99 increases the MTTF from 50 years to 100 years. Thus, when evaluating the results we present, remember that an increase in reliability even as small as 0.01 can be very important.

The experimental results depend greatly on the policies used. For example, Figure 6(b) shows the same results as Figure 6(a), except that the advertising policy <A> has been changed to one that keeps some space in reserve and releases the space over time. (Specifically, this is the *data-proportional* policy with $y = 3$; see Section 3.2.)

Since results depend heavily on the policies used with each algorithm, we have found it useful to graph the "potential maximum" reliability of an algorithm if it were tuned using the best policies. For example, in Figure 7(a), we show the maximum reliability of COLLECTION_TRADING and DEED_TRADING as a function of F_G. To generate each point, we simulated both algorithms with every possible combination of the policies <A>, <S>, <R>, etc., and selected the highest reliability achieved by each algorithm. For example, when $F_G = 3.2$, the best reliability is obtained using the clustering strategy for DEED_TRADING and best fit for COLLECTION_TRADING. In contrast, the points at $F_G = 6$ represent the neighbors strategy for both DEED_TRADING and COLLECTION_TRADING. Thus, Figure 7(a) shows an "upper bound" on the reliability achievable with each algorithm, assuming that the best policies are used. Another way to view these results is shown in Figure 7(b), which shows the differences between the two algorithms by showing the percent difference of DEED_TRADING versus COLLECTION_TRADING. For example, the peak at $F_G = 3.2$ in Figure 7(b) corresponds to the points at $F_G = 3.2$ in Figure 7(a), where DEED_TRADING produces 0.94 global reliability, a nine percent improvement over the COLLECTION_TRADING result of 0.86 global reliability. In this figure, it is clear that DEED_TRADING produces better reliability until $F_G = 4.8$, at which point the reliability of the two algorithms becomes roughly equivalent.

Figure 7(b) shows a dip in the relative reliability at $F_G = 2.4$ that is caused by the uneven increase in global reliability for DEED_TRADING as F_G increases. There are two inflection points in the DEED_TRADING curve in Figure 7(a), such that the rate of increase decreases when $F_G > 2$ and again when $F_G > 3$. Simulation results indicate that at $F_G = 2$ two copies can be made of every collection, while at $F_G = 3$ three copies can be made. At the inflection points increasing space becomes less important; two copies is significantly better than one, three copies is somewhat better than two, but four copies is only slightly better than three. Consequently, "knees" appear in the reliability curve. Such inflection points do not occur for COLLECTION_TRADING because even

when $F_G = 3$, sites have difficulty making three copies of every collection. Thus reaching $F_G = 3$ is not a significant improvement over say $F_G = 2.8$, and the COLLECTION_TRADING curve is more even. The uneven shape of the DEED_TRADING curve, combined with the even shape of the COLLECTION_TRADING curve, produces the dip shown in Figure 7(b).

Our results indicate that DEED_TRADING is the most appropriate algorithm, although if F_G is large, both algorithms are roughly equal. When space is relatively tight (e.g. $F_G = 3$), sites using COLLECTION_TRADING have trouble making three copies of collections. In a trade, the local site must be able to store one of the remote site's collections, but often cannot if space is tight. Thus, no trade can be made, even if the remote site has enough space. Under DEED_TRADING, a trade can always be concluded if the remote site has enough space. If the remote site's collections do not fit in the proposed deed, then the remote site keeps its deed instead of using it. Under DEED_TRADING, the local site can make more trades and almost always makes at least three copies, improving global reliability. As space increases, the impediment to trades under COLLECTION_TRADING disappears, and the algorithm begins to perform as well as DEED_TRADING.

Figure 7 deals with expected reliabilities. (Each value shown is averaged over 100 simulation runs.) So it could be that even though DEED_TRADING has better average performance, it could also have higher variance. This would mean that DEED_TRADING is "riskier" to use, as there could be a few real-life situations where it would provide very poor reliability. We examined the worst reliabilities found in the 100 simulation runs, and found that DEED_TRADING is less "risky" in addition to being better on average. Due to space limitations, this data is presented in the extended version of this paper [10].

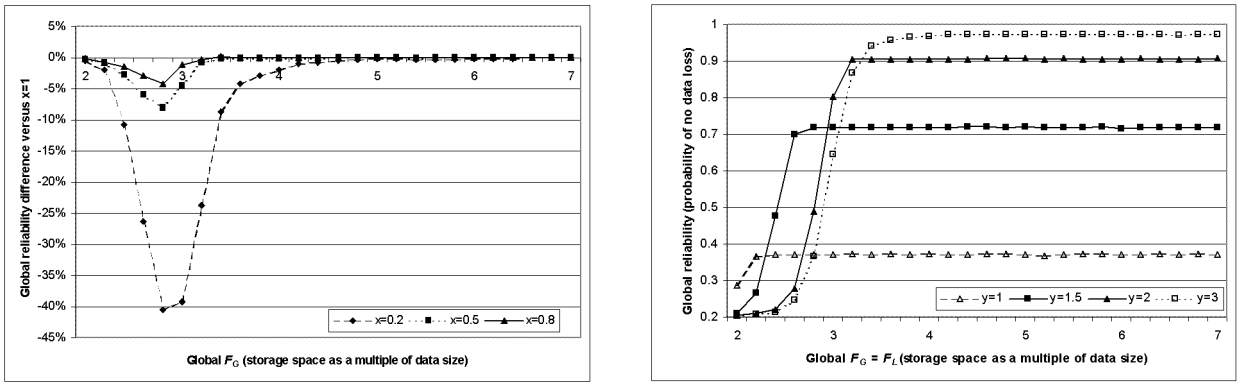
The results we have reported so far assume that all sites have approximately the same space factor. It is possible, and even likely, that different sites will have varying values of F_L. We have examined this situation with an experiment where the local space factors varied in the range $2 \leq F_L \leq 6$, with $F_G = 4$. Thus, in one simulation run, one site might have $F_L = 3.2$ while another might have $F_L = 5.1$. The result is that DEED_TRADING still provides higher reliability, with a global reliability of 0.92, versus 0.88 for COLLECTION_TRADING. Using this same experiment, we have also attempted to measure the "fairness" of each algorithm. If an algorithm were "fair," we would expect the local reliability of the site to increase as the site's F_L increased; thus, sites that contribute more space get correspondingly higher reliability. The results indicate that for both algorithms, sites that have more space experience increased local reliability. Again, this data is presented in [10].

Because the DEED_TRADING algorithm is clearly superior to COLLECTION_TRADING, we report results for DEED_TRADING only for the rest of the paper.

5.2 <A>: The storage space advertising policy

Each site must advertise its free space. A site wishing to make copies of a collection uses these advertisements to determine which sites have enough free space to store a copy. Some of the trading strategies described in Section 3.1 also take the advertised free space into account. Section 3.2 describes the *space-fractional* and *data-proportional* advertising policies. In this section we evaluate these policies.

For simplicity, we have not investigated the process of determining a good partition between local space L_{total} and public space P_{total}. Such a partition must be done carefully, and is related to external factors such as the expected amount of data that will be deposited locally. A site knows the total storage space T it owns locally, and must estimate what ratio F_L of total storage to locally owned collections is needed. We can call this estimate



(a) (b)

Figure 8: The potential maximum reliabilities for the (a) space-fractional and (b) data-proportional policies.

F'_L. Then, $L_{total} = T/F'_L$, and $P_{total} = T - L_{total}$. In our simulation, we have assumed that a good partition has been chosen such that F'_L does indeed represent the final ratio between total storage and locally owned data collections (e.g. $F'_L = F_L$).

The potential maximum reliabilities for the space-fractional policy are shown in Figure 8(a), where $S = 15$ sites, the global space factor F_G varies in the range $2 \leq F_G \leq 7$, and $F_L \approx F_G$ for each site. The figure shows several versions of the space-fractional policy, where $x = 0.8, 0.5$ and 0.2, versus a baseline of $x = 1$. As Figure 8(a) shows, it is much better to advertise all free space (e.g. $x = 1$) if the space-fractional policy is used. Whatever the value of x, all of the public space will eventually be advertised. Therefore, setting $x < 1$ does not adequately reserve space for future use by the local site. On the other hand, if $x < 1$, then remote sites which need to use the public space at the local site may not be able to do so, because the full extent of the free space is not available. As x gets smaller, less space is available at the time of a trade (when the remote sites need the space most). Consequently, the reliability of remote sites suffer, and the global reliability correspondingly decreases.

Figure 8(b) shows potential maximum reliabilities for the data-proportional policy where $y = 1, 1.5, 2$ and 3. (Again, $S = 15$ and $2 \leq F_G \leq 7$.) Each policy peaks when $y = F_L - 1$ and then levels off. This trend continues with $y > 3$, though with diminishing reliability increases. If $y = F_L - 1$, a site will eventually advertise all of its public portion, but only when $L_{used} = L_{total}$. If $y < F_L - 1$, some of the public component is never advertised, while if $y > F_L - 1$ then all of the public component is advertised (and possibly used up) before all local collections have been deposited. Thus, if $y = F_L - 1$, a site uses its public space very efficiently, and always has some space to trade away when new collections are deposited. We simulated the data-proportional policy with $y = F'_L - 1$, and this provided the best global reliability. In order to use this policy, a site must know F'_L, but does not need to know the global F_G. Our experiments indicate that the best global and local reliability results when each site advertises $y = F'_L - 1$ even when F_L varies widely from site to site.

In Figure 9 we compare the space-fractional policy (with $x = 1$) and the data-proportional policy (with $y = F'_L - 1$) directly. Clearly, the data-proportional policy is the best choice. Under the space-fractional policy, a site's public storage can become used up, preventing future trades. Reserved space is released as a result of newly stored remotely owned collections, but there is no guarantee that space will be available for trades. In contrast, with the data-proportional policy, reserved space is released in response to new locally owned collections.

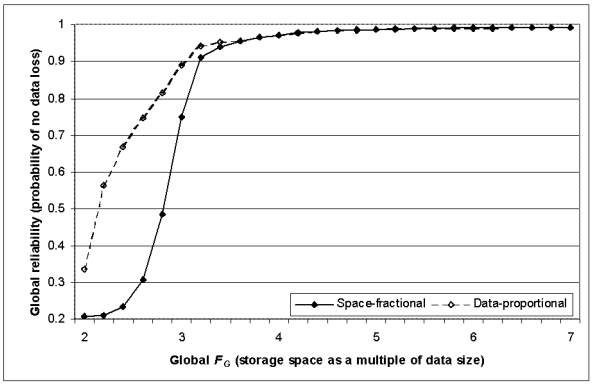
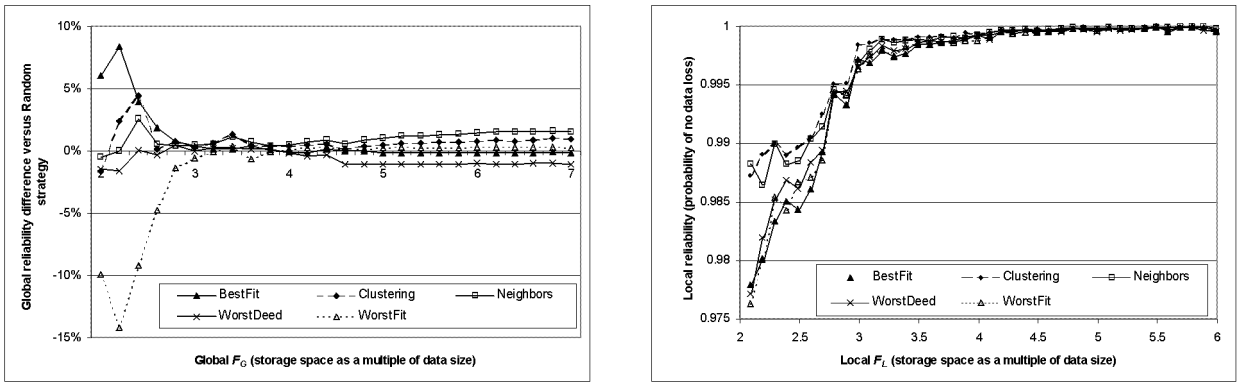


Figure 9: Advertising policies (potential maximum reliabilities).



(a) (b)

Figure 10: Trading strategies: (a) potential maximum global reliabilities, (b) potential maximum local reliabilities.

Consequently, a site looking to make copies of a new collection has new space to trade.

Once again, we have looked at the worst case reliability, the global reliability when F_L varies from site to site, and the impact on local reliabilities. Each of these experiments indicate that data-proportional is the best policy. These results are discussed further in the extended version of this paper [10].

5.3 <S>: Trading strategy

We have implemented the various trading strategies described in Section 3.1 in order to evaluate them. The implementation of most of the strategies was straightforward. For the *neighbors* strategy, each site must choose a local order in which to contact other sites. For our simulation, each site calculates its local order as the older, then younger sites. Thus, the site born third would first contact the site born second, then the site born first, then the site born fourth, and so on.

As shown in Figure 10(a), the best strategy differs depending on the free space factor F_G. This figure shows the difference between each strategy versus a baseline of the random strategy, and again, the potential maximum reliability is reported. For clarity, not all strategies are shown; the omitted strategies are bounded by those in the figure. We also looked at the worst case resulting from each strategy; the results were similar to those shown in Figure 10(b) in that the same strategies provided the best reliability.

When space is tight $F_G \leq 2.8$, best fit provides the highest reliability. In the region $2.8 \leq F_G \leq 4$, the clustering strategy does the best, but for higher values of F_G, neighbors is the best strategy. Clustering is not significantly better than neighbors when $2.8 \leq F_G \leq 4$; however, when $F_G > 4$, neighbors is somewhat better than clustering. Thus it is best to choose neighbors as the strategy when $F_G \geq 2.4$; this is simpler than trying to decide between the two strategies based on F_G. Switching to best fit when space is extremely tight is likely to provide significant benefit. This is because best fit does a good job of matching collections to available space, and minimizing external fragmentation of public storage space. Such fragmentation can be a problem when space is at a premium. Best fit requires global information about the amount of space available at remote sites. Thus, when space is tight, some mechanism of disseminating such information is required, such as through broadcast. As noted earlier, this may contradict the goal of a truly distributed mechanism which preserves site autonomy.

However, as the amount of space increases, fragmentation becomes less of a problem, and sites can make better decisions about clustering their collections together for maximum reliability. Although the "clustering" strategy is designed to achieve clustering, the results suggest that neighbors actually does a better job. Since each site trades preferentially with a locally determined set of neighbors, collections are easily clustered. The "clustering" strategy, on the other hand, must randomly select partners to form clusters with, and such random selection is less successful at forming collection clusters.

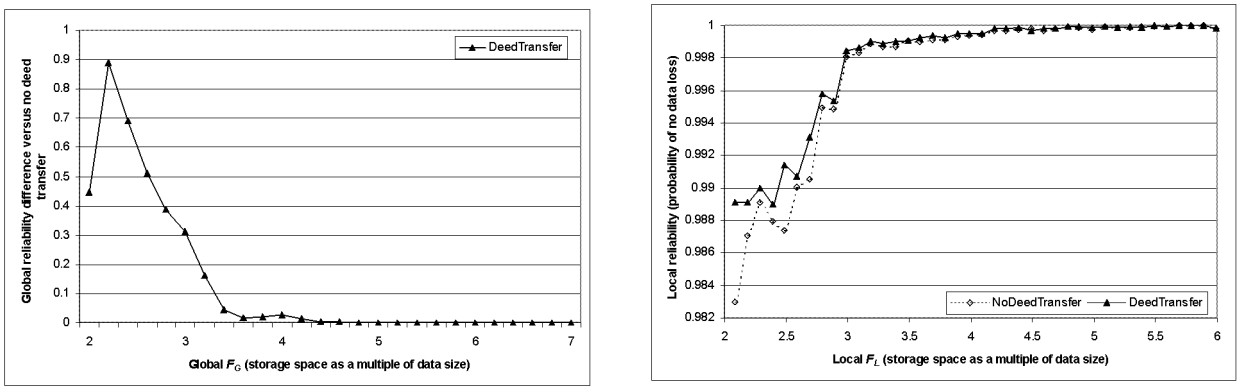
If F_L varies from site to site, then neighbors is still the best strategy. The results from simulations (not shown) with $2 \leq F_L \leq 6$ and $F_G = 4$ show that the neighbors strategy produces a global reliability of 0.923 versus a global reliability of 0.92 for clustering and 0.89 or less for other strategies.

Figure 10(b) shows local reliability versus F_L. We would expect that under a "fair" strategy, sites with more local space experience correspondingly higher reliability. All strategies provide more local reliability for sites with large F_L, although neighbors and clustering provides higher reliability for sites with very little space (e.g. $F_L = 2$). Note that the differences shown in Figure 10(b) may appear to be small but are still significant. For example, the difference between best fit and neighbors when $F_L = 2$ is about 0.01, much larger than the standard deviation in the measurements, which is on the order of 0.002. Moreover, recall from the discussion in Section 5.1 that even a difference on the order of 0.01 can affect the mean time to failure of the system by several decades. Neighbors is the strategy that provides the best global reliability for large values of the global F_G, as indicated in Figure 10(a). When F_G is small (e.g. $F_G = 2$), then best fit is better for global reliability, as shown in Figure 10(a). However, best fit provides the worst local reliability, even for sites with a small local F_L. Therefore, if local reliability is more important than global reliability, sites should use the neighbors strategy.

5.4 <T>: Deed transfer

A site may acquire deeds for other sites that it is unable to use. A site may also have a large amount of effectively free space that is fragmented among several deeds held by other sites. An obvious way to deal with these problems is to allow a site to transfer its deeds to another site. Then, a site wishing to replicate a collection can gather several deeds together into one large enough to hold the collection. For example, if site A wishes to trade with site B, site A may contact other sites that hold deeds for site B. Site A would then trade a deed for its own local space in return for a deed for site B's space.

For simplicity, our simulation assumes that a site will only trade for other existing deeds when the contacted remote site does not have enough undeeded free space remaining. The simulator models a central deed registry,



(a) (b)

Figure 11: The deed transfer policy versus no transfer policy: (a) potential maximum global reliability, (b) potential maximum local reliability.

as described in Section 3.3, which allows a site to atomically acquire as many deeds as it needs. As noted earlier, truly autonomous sites may not wish to use such a central registry, and thus our results represent the best reliability that can be achieved though at the cost of losing some autonomy.

Our results are in Figure 11(a), which shows the potential maximum reliability of deed transfer versus a baseline of no transfer. For example, when $F_G = 3$, allowing deed transfer results in 0.89 global reliability, a 31 percent improvement over the no deed transfer policy (0.68 global reliability). As shown in the figure, the reliability benefit of deed transfer is quite large (up to 90 percent improvement) when storage space is tight, but makes no difference if the free space factor $F_G > 4$. Although the benefit of deed transfer is most pronounced for very low values of F_G, the benefits are still significant when $F_G = 3$: allowing deed transfers results in an absolute global reliability of 0.89 versus 0.68 for no deed transfer. The large benefit clearly suggests that the added complexity of allowing deed transfers is worthwhile. For very large F_G, there is enough space globally so that transfers are not necessary, but allowing them anyway avoids the risk of losing reliability if F_G decreases.

We have also evaluated the situation where each site's F_L differs, such that $2 \leq F_L \leq 6$ and $F_G = 4$. Deed transfer results in a 0.92 global reliability, versus 0.87 for no deed transfer. The local reliability versus the F_L is shown in Figure 11(b). As the figure shows, deed transfer provides higher local reliability for sites with less space (e.g. $F_L = 2$) without harming the local reliability of sites with more space. (Again, the seemingly small differences indicated by the figure can have significant consequences.) This indicates that deed transfer is a good policy, because sites that cannot afford much storage benefit without hurting sites that have a lot of storage.

DEED_TRADING with deed transfer is more complicated to implement than DEED_TRADING without transfer, because a site must have a mechanism to contact third party sites, reserve their deeds, and then acquire their deeds in return for local space. Figure 11(b) indicates that the local reliability of sites with a lot of space does not increase due to deed transfer, and thus sites may try to avoid the complexity of transfer by simply purchasing more disk space. However, unless the global F_G is large, other sites will lose reliability. That is, site A may buy more disk space, but if other sites do not buy extra storage, then the global reliability will be harmed if site A refuses to conduct deed transfers. Our experiments show a 6 percent decrease in global reliability in this situation. As Figure 11(b) shows, it is the sites with less space that will lose reliability. Thus, a site can avoid the complexity of deed transfer but only by harming the reliability of its trading partners.

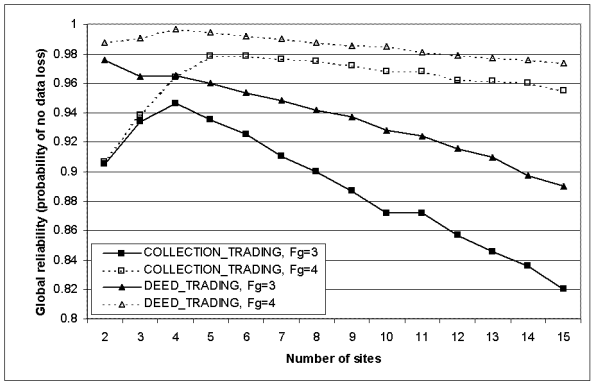


Figure 12: Potential maximum reliability versus the number of sites.

5.5 The impact of the site count

The reliability that can be achieved with data trading varies with the amount of free space available globally as well as the number of sites that are trading. In previous sections we have presented results with F_G, the free space factor, as the independent variable. In this section we discuss the impact of varying S, the number of sites that are trading. The results for both COLLECTION_TRADING and DEED_TRADING, with $F_G = 3$ and $F_G = 4$, are shown in Figure 12. The values shown represent the maximum reliability possible under each algorithm and value of F_G; the strategies and policies used for each data point may differ in practice. As the figure shows, for both algorithms reliability peaks at a relatively small number of sites, and then decreases as more sites enter trading. The value of S at which reliability peaks varies depending on the algorithm as well as the value of F_G. However, it is clear that the best reliability can be achieved when $2 \leq S \leq 9$, and that if more than nine sites enter the trading, then reliability suffers.

Global reliability suffers when there are too few sites because sites are unable to make sufficient trades. When there are only two sites, for example, one site may have a large data collection but be unable to replicate it at all if the second site has insufficient storage. As more sites enter trading, the chance of finding a site with enough space to store a copy of a collection increases.

However, global reliability peaks and then decreases with increasing site count. As more sites enter the trading network, there are more collections competing for space. Although the global aggregate amount of space increases with increasing site count, the space available at local sites remains fixed, as a function of the local site factor F_L. Consequently, as more collections compete for space, it is more likely that one or more sites will find their local public storage filled up. These sites cannot conduct trades, and the decrease in their local reliability drags down the global reliability. When the number of sites is small, this effect lessens, and higher global reliability results.

This result indicates that it is best to form partnerships of a few sites and to trade within these partnerships rather than trying to include all archiving sites in the same trading network. In fact, this is a positive result; it shows that a very large number of sites is not needed to achieve reliability, and that a small confederation of archives can trade among themselves to achieve high reliability. Moreover, it indicates that data trading scales well to a very large number of sites simply by dividing the sites into small groups.

Space is tight, there are few sites	Algorithm: Use COLLECTION_TRADING <A>: Use the data proportional policy, with $y = F'_L - 1$ <R>: Active retries should be disallowed <S>: Use neediest
Space is tight, there are many sites	Algorithm: Use DEED_TRADING <A>: Use the data proportional policy, with $y = F'_L - 1$ <R>: Active retries should be disallowed <S>: Use best fit <T>: Deed transfer is significantly beneficial <U>: Non-aggressive deed use is best
Space is adequate, there are any number of sites	Algorithm: Use DEED_TRADING <A>: Use the data proportional policy, with $y = F'_L - 1$ <R>: Active retries should be allowed <S>: Use neighbors <T>: Deed transfer is somewhat beneficial <U>: Non-aggressive deed use is best
Space is plentiful, there are any number of sites	Algorithm: Use DEED_TRADING <A>: Use the data proportional policy, with $y = F'_L - 1$ <R>: Active retries should be allowed <S>: Use neighbors <T>: Deed transfer is does not help or hurt <U>: Aggressive deed use is best

Figure 13: Summary of the most reliable policies

5.6 Summary

Our experiments (presented in this paper and in the extended version [10]) indicate that the most appropriate trading mechanism has the characteristics shown in Figure 13. This figure divides the policies into the situations where space is tight (e.g. $F_G = 2$), space is adequate (e.g. $F_G = 3$), and space is plentiful (e.g. $F_G = 5$). The figure also illustrates that different policies are best if the number of sites is small (e.g. $S = 6$) and space is tight (e.g. $F_G = 2$).

We have also conducted experiments where the reliability varied from site to site. Our results show that the policies in Figure 13 still apply, with two exceptions. First, highly reliable sites do better by offering trades to other highly reliable sites, but less reliable sites should still try to trade with all sites, regardless of reliability. Second, highly reliable sites can benefit by asking for larger deeds from less reliable sites, with a corresponding detrimental effect for the less reliable sites. We do not have the space to report more complete results here; see [9].

6 Related work

The problem of optimally allocating data objects given space constraints is well known in computer science. Distributed bin packing problems [24] and the File Allocation Problem [7] are known to be NP-hard. This is one reason we have not sought to find an optimal placement for data collections. Moreover, these problems are even harder when the number of sites and number and sizes of collections are not known in advance.

Existing schemes for data replication focus as much on access performance and load balancing after a failure as on surviving the failures themselves [11, 29, 30]. In Section 3.1, we presented two possible configurations for distributing four collections among four sites. These configurations are analogs of disk array architectures studied by other researchers; configuration I is equivalent to a mirrored disk architecture [6], while configuration II is

known as chained declustering [20]. Previous research suggests that configuration II is usually preferred despite its lower reliability, because after a failure, the increased load on the surviving sites is better distributed [22]. In our model, reliability is more important than performance, so we would prefer configuration I (e.g. mirrored disks) if possible. However, due to the dynamic nature of the system, with new sites and new collections appearing at any time, it is not possible to statically assign copies to mirrored disk pairs, and a more dynamic data allocation scheme must be devised.

Data replication schemes must also deal with updates to the data. For this reason, much work has been done to ensure correctness and consistency for distributed transactions [4, 18]. Our work focuses on placing data in the most reliable manner in an archival setting. We assume that archived data is not updated. Update consistency protocols could be used in addition to our trading protocol if necessary.

Replication schemes such as Coda [21] or Andrew [13] use caching to improve availability. Data is replicated so that it can be read despite temporary site failures or network partitions. This is a different goal than long term reliability, which means preserving data despite permanent site failures or data corruption. 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 replicates are not ejected once placed.

Digital library researchers have begun to examine the archiving problem [16]. Some projects have focused on maintaining collection metadata [26], or on dealing with the data formats [28, 23, 19]. Others have focused on policy issues surrounding information archiving [5]. Each of these issues are important, and complement the basic bit-level reliability we seek to provide here. Projects such as Intermemory [17], SAV [8], LOCKSS [27], or OceanStore [14] do focus on bit-level reliability, and these efforts must effectively replicate information to different sites. Our trading algorithm could be used in systems like these to place data in the most reliable manner.

7 Conclusions

We have proposed a new way to distribute copies of data collections among cooperating data storage sites. By trading in a peer to peer manner, sites can make multiple copies of their collections despite a highly dynamic, distributed environment. In this way, archives can ensure that their data is reliably preserved. We have defined basic algorithms for data trading, and identified policies that can be used to tune these algorithms. We have also proposed several alternative choices that could be made for each of these policies.

Using our simulator, we have evaluated these alternative choices. For some decisions, a clear policy should always be followed; for example, a site should always use the data-proportional policy when advertising its free space to other sites. For other decisions, the best policy depends on the amount of free space available in the system. Usually, the policy that appears to be the best under a criterion of global reliability is also best from the perspective of local reliability, although the deed use policy is an exception (see [10]). Moreover, we have determined that the best reliability is achieved when just a few sites (e.g. less than 10) trade together. These conclusions can be used to design a peer to peer data trading protocol among any collection of storage sites interested primarily in reliability.

References

[1] The Freenet Project. http://freenet.sourceforge.net/, 2000.

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

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

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

[5] N. Beagrie. Developing a policy framework for digital preservation. In *Proc. of the Sixth DELOS Workshop on Preservation of Digital Information*, June 1998.

[6] A. Borr. Transaction monitoring in Encompass [TM]: Reliable distributed transaction processing. In *Proc. 7th VLDB*, Sept. 1981.

[7] W. W. Chu. Multiple file allocation in a multiple computer system. *IEEE Transactions on Computing*, C-18(10):885–889, Oct. 1969.

[8] 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.

[9] B. Cooper and H. Garcia-Molina. Creating trading networks of digital archives. http://dbpubs.stanford.edu/pub/2001-4, 2001. Technical Report.

[10] B. Cooper and H. Garcia-Molina. Peer to peer data trading to preserve information (Extended version). http://-dbpubs.stanford.edu/pub/2001-6, 2001.

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

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

[13] J. H. Morris et al. Andrew: A distributed personal computing environment. *CACM*, 29(3):184–201, March 1986.

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

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

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

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

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

[19] A. Heminger and S. Robertson. Digital Rosetta Stone: A conceptual model for maintaining long-term access to digital documents. In *Proc. of the Sixth DELOS Workshop on Preservation of Digital Information*, June 1998.

[20] H. Hsiao and D. DeWitt. Chained declustering: A new availability strategy for multiprocessor database machines. In *Proc. 6th ICDE*, Feb. 1990.

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

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

[23] N. Maria, P. Gaspar, A. Ferreira, and M. Silva. Information preservation in ARIADNE. In *Proc. of the Sixth DELOS Workshop on Preservation of Digital Information*, June 1998.

[24] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. J. Wiley and Sons, Chichester, New York, 1990.

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

[26] A. Rajasekar, R. Marciano, and R. Moore. Collection-based persistent archives. http://www.sdsc.edu/NARA/-Publications/OTHER/Persistent/Persistent.html, 2000.

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

[28] J. Rothenberg. Ensuring the longevity of digital documents. *Scientific American*, 272(1):24–29, Jan. 1995.

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

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