Peer-to-peer resource trading in a reliable distributed system

Brian F. Cooper and Hector Garcia-Molina

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

Abstract

Peer-to-peer architectures can be used to build a robust, fault tolerant infrastructure for important services. One example is a peer-to-peer data replication system, in which digital collections are protected from failure by being replicated at multiple peers. We argue that such community-based redundancy, in which multiple sites contribute resources to build a fault-tolerant system, is an important application of peer-to-peer networking. In such a system, there must be flexible, effective techniques for managing resource allocation. We propose data trading, a mechanism where a site acquires remote resources in the community by trading away its own local resources. We discuss the application of data trading to the data replication problem, and examine other applications of trading. A general trading infrastructure is a valuable part of a peer-to-peer, community-based redundancy system.

1 Introduction

Peer-to-peer systems form a useful architecture for a wide range of important applications. Although the term "peer-to-peer" is often associated in the public imagination with Napster and related file-sharing systems, other important services that can be built on a peer-to-peer framework. For example, a group of digital libraries may cooperate with each other to provide preservation by storing copies of each other's digital materials. In this system, each library acts as an autonomous peer in a distributed, heterogeneous collection replication mechanism. Such a community does not require a central controller to manage the replication of data; instead, each peer can communicate with other peers to replicate its own collections. The result of individual libraries seeking locally to preserve their own information by working with other peers is a global community in which every library's collections are protected.

Such a replication network is an example of a *community-based redundancy system*: a group of peers collaborate to provide resource redundancy and thus reliability and fault tolerance. There are several benefits to community-based redundancy. First, each peer is able to take advantage of a system with large aggregate resources simply by contributing its own, relatively small set of resources. Second, the distribution of resources in the system means that the value of the aggregate resources is larger than the sum of the individual contributions. For example, it is more valuable to the Stanford library to have one copy of its collections locally and one copy at say MIT than it is for Stanford to have two copies locally. If Stanford experiences a failure (such as a hardware fault, a malicious attack such as a virus or trojan, or a natural disaster), then it can recover from the failure by using the unaffected copy at MIT. Third, the heterogeneity inherent in a community of autonomous sites is valuable; if all sites are homogeneous then a software bug or security vulnerability that afflicts one site would afflict all sites. Because of these advantages, several systems have been built on the model of community-based redundancy, including Archival Intermemory [6], OceanStore [17], LOCKSS [3], and SAV [9].

A central question in community-based redundancy systems revolves around the contribution and allocation of resources. Peers must determine how many resources they can reasonably expect from the community, and how many resources they themselves must contribute. Moreover, because there is no central allocation mechanism, participants must make careful decisions when determining how resources are used, in order to avoid a situation where the needs of some participants are not met by the community despite the fact that there are nominally enough resources available to meet everyone's needs.

In order to deal with these allocation issues, we are investigating a mechanism that we call *data trading*. One application of data trading is digital archiving, where sites protect their collections from failures by making multiple copies at remote sites. When a site has a digital collection it wishes to replicate, the site contacts a remote site and proposes a trade. For example, Stanford's library may have a collection of technical reports that it wants to preserve, and thus Stanford contacts MIT and proposes a trade. MIT might respond that it is willing to store Stanford's collection, if in turn Stanford is willing to store a copy of a collection of scientific measurement data owned by MIT. A series of such binary trades creates a peer-to-peer trading network. Each peer tries to maximize its own local reliability, but the effect is that the whole network is a reliable infrastructure for archiving data.

A trading-based peer-to-peer system has several advantages. First, it preserves the autonomy of individual peers. Each site makes local decisions about who to trade with, how many resources to contribute to the community, how many trades to try to make, and so on. Sites are more willing to participate in a peer-to-peer scheme if they can retain this local decision making. Second, the symmetric nature of trading ensures fairness and discourages free-loading. In order to acquire resources from the community, each peer must contribute its own resources in trade. Moreover, sites that contribute more resources receive more benefit in return, because they can make more trades. Third, the system is robust in the face of failure. Because the trading network is composed of myriad binary trading links, individual links or sites can fail without crashing the whole network. Instead, the "broken" trading links can be replaced with freshly negotiated links between surviving peers.

In this position paper, we argue that these advantages make trading a useful component of the infrastructure of peer-to-peer systems and of community-based redundancy systems based on a peer-to-peer architecture. Specifically, we describe our current work in developing algorithms and policies for trading, and then discuss how the generality of a trading mechanism makes it widely applicable to a variety of problems.

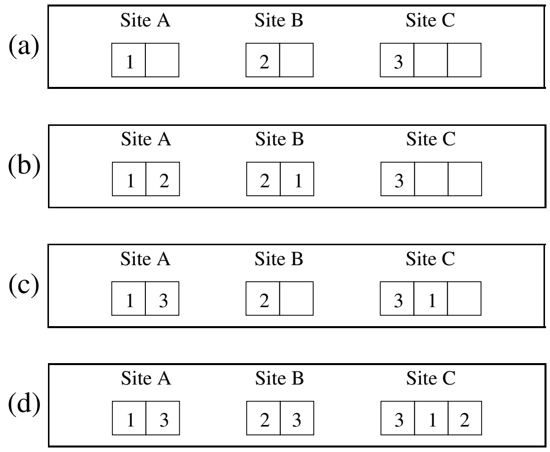

Figure 1: Data trading example.

2 Data trading

In our current work, we have focused on trading for the application of reliable preservation through replication. The basic framework for trading in this application can be tuned in different ways to achieve the highest reliability. In this section, we illustrate an example of a trading session and then discuss the reliability optimizations we have studied.

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.

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.

This example illustrates that a great deal of care must be put into the local decisions that are made by each site. Although it is often impossible to make optimal decisions, especially without knowledge of future events (such as new sites, new storage space added to an existing site, etc.) we can study useful heuristics that tend to improve the overall reliability of the system. We can encapsulate these heuristics in *trading policies* that guide the local decision making at each peer. In this way, the continuous process of offering and accepting trades can be automated. The system, once configured with appropriate trading policies, autonomously replicates information to ensure high reliability.

2.1 Trading policies

We have studied several different policies that determine the the behavior of a trading peer. Examples of policies include:

Deed trading. One possibility is for peers to trade collections directly, and this is the approach assumed in the example presented above. This approach has the disadvantage that trades may not be very symmetric; for example, Stanford's collection may be much larger than MIT's, which results in a situation where MIT gives away more storage than it gets in return. A fairer scheme (and, it turns out, more reliable scheme) is one in which blocks of space are traded. For example, Stanford may give MIT 10 GB of space and in return get 10 GB of MIT's space. Each site can then use the space it has acquired as it sees fit. If MIT's collection is smaller than 10 GB, it may be able to use the space it acquired at Stanford to replicate several collections. The bookkeeping mechanism for tracking these trades is called *deeds*: a deed represents the right to use space at another site. Once MIT has acquired a deed for space at Stanford, it can use the deed, save it for the future, split it into smaller pieces, or trade the deed away to another site, as it sees fit.

Advertising policy. A site advertises the amount of storage space it is willing to trade away. In the simplest case, a site advertises all of the space it has free. However, higher reliability over the long term can be achieved by reserving some space for future use. Then, a site only advertises a fraction of its available resources at any one time. This ensures that there is always some space to trade away, which may be needed if the site gets a new collection that it must replicate by proposing new trades.

Remote site selection strategy. When a site wishes to make trades, it must decide which remote sites to offer trades to. One possibility is to choose the remote site that has the lowest probability of failure, estimated based on previous history, reputation, or the quality of components at the site. However, this policy is counterproductive if every peer uses it, because the "high reliability" sites quickly become overloaded. A more effective policy is for peers to choose a small set of "trusted" trading partners, and trade repeatedly with those partners.

Bidding policy. When Stanford asks MIT for a trade, the two sites may exchange equally sized blocks of space. An alternative is for Stanford to offer a trade by saying that it needs a certain amount of space at MIT (say, 10 GB), and asking how much space MIT would want in return. If MIT is eager to trade, because it has many collections to replicate, it may offer a low *bid*, asking for only 5 GB in return. On the other hand, if MIT is reluctant to trade, say because its local space is becoming scarce, then it may offer a high bid, asking for 15 GB in return. In this way, Stanford can contact multiple sites, get bids from each one, and then accept the most attractive offer. In this scenario, each site must decide, based on its local circumstances, what bid to offer for each trade.

These and other policies have been examined in more detail in [11, 12, 10]. These papers also describe our trading simulator, a system we have built to simulate trading sessions using different policies. Our simulator has allowed us to identify policies which provide the highest reliability in different circumstances.

3 Generalizing trading in peer-to-peer systems

Although we have studied data trading specifically in the context of trading storage space to replicate collections, we believe it is a general mechanism for several applications. Trading can serve as a part of the infrastructure of a peer-to-peer, community-based redundancy system. In this section we outline some other possible uses for trading.

Trading can be used to exchange resources besides storage space. For example, processing cycles for searching collections can also be traded. Once collections are distributed in a community-based replication system, users will attempt to find collections or individual documents within collections by performing searches. In a highly reliable, highly available system, users should still be able to perform searches even in the presence of site failures. If one site is charged with handling searches for a particular collection, and that site fails or is unreachable due to a network partition, then users will not be able to search the collection even though copies may still be available in the network. On the other hand, the search load may be replicated and distributed in the same way that the physical collections are replicated. If Stanford agrees to process searches over MIT's collections, and in return MIT agrees to process searches over Stanford's collections, then collections can always be searched despite a failure at either site.

In addition to trading processing for processing, it is possible that a trading infrastructure can be used to trade one type of resource for another. For example, the site that "owns" a collection may not have the processing capacity or bandwidth to support all of the searches submitted by users. On the other hand, this site may have an excess of storage space. The site may try to shed some of the query load by contracting with other sites that will take over some of the search processing. The mechanism for contracting may be based on trading, in which the site gives away some of its storage space in return for processing cycles at other sites.

Trading may also be extended for exchanging more abstract resources. One example is trading access to content. A site may have a limited budget, and is not able to directly purchase access to important collections. However, that site may also have collections of its own that are desired by other sites. Then, the site can gain the right to use a valuable collection owned by another site by trading away the right to use its own collections.

In general, any redundancy systems that allocate limited resources can use a trading mechanism as an infrastructure component. Some existing systems allocate redundant resources in a fixed, static way. Although it is possible to reason about good or even optimal policies for certain configurations, it is difficult to do so in a distributed system with autonomous peers. Moreover, if the configuration is highly dynamic then the fixed allocation may no longer be appropriate. In contrast, other existing distributed and peer-to-peer systems allocate resources in response to user demand, or even randomly. Allocating in response to user requests may mean that less popular collections are not preserved at all. Allocating randomly may make inefficient use of community resources. If the goal is to ensure redundancy and high reliability, then trading provides a way to achieve effective allocation while dynamically adapting to changes in user requirements and network configuration.

Finally, in a general trading system, it may be difficult to distinguish trustworthy trading partners from less reliable or malicious sites. In some cases, it may be sufficient to implement a reputation system to identify (and ostracize) peers that do not fulfill their responsibilities. However, reputation systems only operate after a node has misbehaved. It may be necessary to implement a security policies to prevent or at least mitigate malicious attacks. Preliminary work in this area is described in [8].

3.1 Related work

Our work draws upon concepts developed in related systems. Traditional data management schemes, such as replicated DBMS's [5, 18], replicated filesystems [15] and RAID disk arrays [22] utilize replication to protect against failures in the short term. A peer-to-peer trading system provides more autonomy and fairness for individual storage components than traditional solutions. Another difference is that traditional solutions are concerned with load distribution, query time and update performance, as well as reliability [14, 23, 24]. Here, we are primarily concerned about preservation (given the constraint of preserving site autonomy). In contrast, traditional replicated databases tend to trade some reliability for increased performance [20]. Similarly, replicated filesystem schemes such as Coda [19] or Andrew [16] use caching to improve availability. Our goal is different: long term preservation despite failures, rather than short term preservation in the face of network partitions.

Many existing peer-to-peer such as Freenet [1] or Gnutella [2] use "trading" as a model, although these systems trade content (such documents or audio files), not necessarily resources (such as storage). Also, these 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.

Other peer-to-peer systems have focused on reliable storage using an economic model similar to trading. FreeHaven [13] uses a trading system very similar to our work. However, anonymity and peer accountability are primary goals of FreeHaven. As a result, trades occur in order to obscure the true owner of a document, and trading partners are chosen based on reputation. Our model aims for a different goal, that of long term reliability; at the same time, we examine (and scientifically evaluate) a wider range of policies for choosing trading partners. MojoNation [4] also used trading, but transactions were made via an intermediate currency called "mojo." As with any currency-based mechanism, the system is vulnerable to fluctuations in the money supply and manipulations of the currency value. Barter, such as in our work, attempts to avoid these problems.

Systems such as the Archival Intermemory [6] and OceanStore [17] are very good at preserving digital objects through replication. Our trading techniques could serve as the storage allocation and replica placement mechanism for these systems, increasing reliability and providing site autonomy.

The problem of optimally allocating data objects given space constraints is well known in computer science. Distributed bin packing problems [21] 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.

4 Conclusion

A peer-to-peer infrastructure is useful for a variety of applications. One important application is reliability through redundancy. The large number of individual resources in the community, the geographical and administrative distribution, and the heterogeneity of peers are all advantages to a peer-to-peer architecture for community-based redundancy system. In such a system, it is vital that peers can use a dynamic, flexible and effective mechanism for allocating resources. Data trading provides such a mechanism. Because it preserves autonomy, ensures fairness and is robust in the face of failure, trading is a good mechanism for providing community-wide replication through decisions made locally by sites in their own self interest.

We have focused on trading as a way to allocate storage space for data replication. Our research has examined several heuristic policies that can be used to achieve high reliability in such a trading system despite the limited information available to each local site. We have also argued that the trading framework can be extended for other purposes and applications. The advantages of trading make it a valuable component of the peer-to-peer infrastructure.

References

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

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

[3] Lots of copies keeps stuff safe (LOCKSS). http://-lockss.stanford.edu/, 2001.

[4] MojoNation. http://www.mojonation.net/, 2002.

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

[6] Yuan Chen, Jan Edler, Andrew V. Goldberg, Allan Gottlieb, Sumeet Sobti, and Peter N. Yianilos. A prototype implementation of archival intermemory. In *Proc. ACM Int'l Conf. on Digital Libraries*, 1999.

[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. F. Cooper, M. Bawa, N. Daswani, and H. Garcia-Molina. Protecting the PIPE from malicious peers. http://www-db.stanford.edu/~ cooperb/pubs/-pipe.pdf, 2002. Technical report.

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

[10] B. F. Cooper and H. Garcia-Molina. Bidding for storage space in a peer-to-peer data preservation system. http://dbpubs.stanford.edu/pub/2001-52, 2001. Technical Report.

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

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

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

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

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

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

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

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

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

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

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

[22] 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.

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

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