# Redundancy Schemes for High Availability in DHTs

Fan Wu, Tongqing Qiu, Yuequan Chen, and Guihai Chen

State Key Laboratory of Novel Software Technology,
Nanjing University, China

**Abstract.** High availability in peer-to-peer DHTs requires data redundancy. This paper takes user download behavior into account to evaluate redundancy schemes in data storage and share systems. Furthermore, we propose a hybrid redundancy scheme of replication and erasure coding. Experiment results show that replication scheme saves more bandwidth than erasure coding scheme, although it requires more storage space, when average node availability is higher than 48%. Our hybrid scheme saves more maintenance bandwidth with acceptable redundancy factor.

**Keywords:** Peer-to-Peer, Distributed Hash Table, Replication, Erasure Coding.

## 1 Introduction

The last several years have seen the emergence of a class of structured peer-to-peer systems that provide a distributed hash table (DHT) abstraction [13, 16, 18]. DHTs propose a determined object locating service, and already are used in many applications [6, 7, 10, 17]. However, to provide high data availability in the DHT, when the peers that are storing them are not 100% available, needs some form of data redundancy. Peer-to-peer DHTs have proposed two different redundancy schemes: replication [6, 17] and erasure coding [7, 10].

Some comparisons [2, 7, 19] argued that erasure coding is the clear winner, due to huge storage and bandwidth savings for the same availability levels (or conversely, huge availability gains for the same storage space). The other comparisons [3, 15] argued that coding is an clear winner only when peer availability is low; the benefits of coding are so limited in some cases that they can easily be outweighed by some disadvantages such as extra complexity, download latency and lack of ability of keyword searching.

This paper argues that sharing user downloaded files for subsequent accesses (replication) and meanwhile utilizing erasure coding to maintain files' availability will achieve better performance: saving more bandwidth with acceptable redundancy factor. There are two talking points. First, in current peer-to-peer file sharing communities, popular files are automatically kept at high availability level, due to thousands of times of user downloads. Second, current hardware deployment suggests that idle bandwidth is the limiting resource that volunteers contribute, not idle disk space. Further, since disk space grows much faster than access point bandwidth, bandwidth is likely to become even scarcer relative to disk space.

This paper makes the following contributions: First, to our best knowledge, this paper is the first to take user download behavior into account — sharing user downloaded files for subsequent accesses — to evaluate redundancy schemes in data storage and share systems. Second, this paper demonstrates that replication saves more bandwidth than erasure coding, although it requires more storage space, when average peer availability is higher than 48%. Finally, this paper shows that the hybrid redundancy scheme of replication and erasure coding can achieve better overall performance: saving more bandwidth with redundancy factor less than 9.4 for three nines (99.9%) of per-file availability.

The rest of the paper is organized as follows. Section 2 formulates and describes three schemes for high availability: replication, erasure coding and the combination of them. Section 3 evaluates these three schemes by two sets of experiments. Finally, we conclude the paper and point out future work in Section 4.

## 2 Redundancy Schemes

This section presents three redundancy schemes for high availability: replication, erasure coding and a hybrid scheme which shares user downloaded files for subsequent accesses (replication) and utilizes erasure coding to adjust files' availability. All of them work upon consistent hashing [9], as used by storage systems such as CFS [6].

First, several key terminologies should be introduced. For simplicity, each file is identified by a unique identifier $d$, which is consistent hash of the file name. The peer that keeps location indexes for file copies or fragments is named *indexer*. Besides, a dualistic hash function should be declared: $h(d, n)$, where $n \geq 1$ is the sequence number of each indexer. $h(\bullet, \bullet)$ is the allocation function, typically based on the hash function shared by all peers. The allocation function might be defined as follows: $h(d, n) = H(d \,||\, n)$, where $H(\bullet)$ is the hash function which is used in the DHTs and $||$ is a concatenation.

All schemes consist of three parts with some difference due to their particularity: register, request and maintenance.

- Register: Each peer periodically registers the unique IDs of the files it holds and/or fragments in its cache in $M$ distributed and independent indexers. The logical location of M indexes is determined by the hash function defined above: $h(d, n), n \in [1, M]$. If the peer pointed by $h(d, n)$ is not alive, its successor takes over its role. The indexer associates each item in the index with a timer. A copy of file or a fragment will be recognized as unavailable and removed from the index if its timer runs out.
- Request: When requesting a file $d$, a peer randomly refers to one or more indexers responsible for $d$. If the checked indexers do not provide enough whole file or fragment location information, the peer will turn to other indexers. If all $M$ distributed indexers fail to provide enough location information, the peer will wait a period of time and do the procedure as stated above again, until maximum lookup time expires. This balances the load of directory service and reduces the chance of getting incomplete location index. Then the peer downloads the file or enough fragments to reconstruct the original file from peers registered in location index.
- Maintenance: Periodically, each indexer estimates the availability of files and/or fragments registered on it, and attempts to increase the availability of ones that is not yet at target availability.

## 2.1 Replication Scheme

Replication is the simplest redundancy scheme. Here $r$ identical copies of each file are kept at each instant by peers. The value of $r$ must be set appropriately depending on the desired object availability $a$ (i.e., $a$ has some "number of nines"), and on the average peer availability $p$. Throughout the analysis, it is assumed that the peer availability is independent and identically distributed. The needed number of copies can be determined by:

$$a = 1 - (1 - p)^r \tag{1}$$

which upon solving for $r$ yields

$$r = \frac{\log(1 - a)}{\log(1 - p)} \tag{2}$$

Each peer periodically registers shared and cached files in $M$ distributed and independent indexes. When requesting a file $d$, a peer lookups a random index responsible for $d$. If the referred indexer fails, the peer will turn to another indexer. If all $M$ indexers fail, the peer will wait a period of time and do the lookup procedure again, until maximum lookup time is reached. Then the peer accesses the file from a random peer registered in location index. The already downloaded file is automatically treated as a shared file for subsequent accesses. Finally, each indexer periodically adjust the availability of its indexed files by scheduling necessary number of file transfers from the whole file holder to randomly chosen peers to reach the desired availability of file $d$.

## 2.2 Erasure Coding Scheme

Erasure codes (e.g., Reed-Solomon [14] or Tornado [5]) divide an object into $m$ fragments and recode them into $n$ fragments, where $n > m$. This means that the effective redundancy factor is $r = n/m$. The common property of erasure codes is that the original object can be reconstructed from any $m$ fragments (where the combined size of $m$ fragments is approximately equal to the original object size).

We assume that we place one encoded fragment per file per peer and there is no duplicate fragments. File availability can be calculated by the probability of at least $m$ out of $n$ fragments are available:

$$a = \sum_{i=m}^{n} \binom{n}{i} p^i (1 - p)^{n-i} \tag{3}$$

where $p$ is the average peer availability.

The number of files per host follows a Poisson distribution. Because it is difficult to directly evaluate the Poisson distribution, we use the normal approximation to the Poisson distribution. With the normal approximation, if we perform random placement of files on hosts then the number of files per host follows a normal distribution. Using algebraic simplifications and the normal approximation to the binomial distribution (see [1]), we get the following formula for the erasure coding redundancy factor:

$$r = \frac{n}{m} = \left( \frac{\sigma_a \sqrt{\frac{p(1-p)}{m}} + \sqrt{\frac{{\sigma_a}^2 p(1-p)}{m} + 4p}}{2p} \right)^2 \tag{4}$$

where $\sigma_a$ is the value of standard deviations in a normal distribution for the required level of availability. Table 1 shows the standard deviations in a normal distribution for different values of availability $a$. These results are standard for any normal distribution. For instance, $\sigma_a = 3.1$ corresponds to three nines of availability.

**Table 1.** Standard deviations that follows a normal distribution for the given level of availability

| $a$ | $\sigma_a$ |
|---|---|
| 0.800 | 0.84 |
| 0.900 | 1.28 |
| 0.990 | 2.48 |
| 0.995 | 2.81 |
| 0.998 | 2.88 |
| 0.999 | 3.10 |

When erasure coding is used, an indexer that generates new fragments to adjust availability must have access to the whole file. It is unscalable to download enough fragments to reconstruct the file and then generate new fragments, since it is likely that $m$ fragments need to be downloaded to regenerate merely a new fragment. Thus the amount of file that needs to be transferred is $m$ times as much as the amount of redundancy lost. An alternative is to associate the peer whose identifier is closest to the consistent hash of the file name as the *home peer* for that file. The home peer stores a permanent copy of the file and manages its fragment generation. If the home peer fails, the next closest peer in the identifier space automatically becomes the new home peer. This is reasonable because the peer that takes responsibility of a file restores a complete copy, generates and pushes new fragments to targets in need. This corresponds to increasing the redundancy factor by 1.

However, erasure coding scheme does not share the whole user downloaded files. All shared objects in the system are erasure coded fragments stored in caches. Each peer periodically registers all fragments it keeps in $M$ distributed and independent indexes. When requesting a file $d$, a peer lookups a random indexer responsible for $d$'s fragments. If the referred indexer can not provide enough fragment location information, the peer will turn to another indexer. If all $M$ indexers fail, the peer will wait a period of time and do the lookup procedure again, until maximum lookup time is reached. Then it downloads enough number of fragments and resembles the original file, and tries to regenerate and leave a fragment in cache. Finally, each indexer periodically adjusts the availability of its indexed fragments. For a file whose availability is below target level, the indexer consigns the home peer of the file to generate and push necessary number of fragments to randomly selected peers.

## 2.3 Hybrid Scheme

The replication scheme shares user downloaded files for subsequent accesses to save maintenance bandwidth. It saves more maintenance bandwidth than the erasure coding scheme when average peer availability is high, but requires much larger redundancy factor. The erasure coding scheme requires much less storage space than the replication to reach the availability level with the same average peer availability, and saves more maintenance bandwidth in highly dynamic environment, but still unscalable when average peer availability is low. This paper proposes a hybrid scheme which combines replication and erasure coding to achieve to better overall bandwidth saving with acceptable redundancy factor.

The hybrid scheme shares user downloaded files for subsequent accesses (replication) and utilizes erasure coding to maintain files' availability. It automatically treats a downloaded file as shared file for subsequent accesses as the replication scheme. When adjusting file availability, it consigns a whole file holder to generate and push necessary number of fragments to other peers, instead of transferring whole copy of file. On one hand, the hybrid scheme utilizes file copies already downloaded on network for subsequent downloads to reduce maintenance bandwidth overhead as the replication scheme. On the other hand, the hybrid scheme uses erasure coding to achieve less bandwidth overhead than replication for the same increment of availability level.

We now exhibit the analogue of Equation (1) and (3) for the case of hybrid scheme. We assume that we do not place files and fragments with the same ID on the same peer, and there is no duplicate fragments. File availability $a$, can be calculated by the probability of at least a whole copy or at least $m$ out of $n$ fragments are available. So $a$ is estimated as 1 minus the probability that all whole copies of a file are simultaneously unavailable and there are not enough (at least $m$ out of $n$) fragments available to reconstruct the original file:

$$a = 1 - (1-p)^h \left(1 - \sum_{i=m}^{n} \binom{n}{i} p^i (1-p)^{n-i}\right) \tag{5}$$

where $h$ is number of file copies.

The hybrid scheme's redundancy factor can be calculated by adding redundancy factor of replication and erasure coding:

$$r = h + \frac{n}{m} = h + \left(\frac{\sigma_a(h)\sqrt{\frac{p(1-p)}{m}} + \sqrt{\frac{\sigma_a{}^2(h)p(1-p)}{m} + 4p}}{2p}\right)^2 \tag{6}$$

where $\sigma_a(h)$ is a function of $h$, and its value corresponds to the availability level (see Table 1) $a'$ that erasure coding has to obtain. $a'$ is derived from Equation (5) as follows:

$$a' = \sum_{i=m}^{n} \binom{n}{i} p^i (1-p)^{n-i} = 1 - \frac{1-a}{(1-p)^h} \tag{7}$$

Figure 1 captures the theoretical redundancy factor for the replication, erasure coding and hybrid schemes determined by Equation (2), (4) and (6) to achieve three nines of per-file availability. The redundancy factor of the hybrid scheme is determined by two factors: average peer availability $p$ and number of file copies $h$. With any fixed $h$, there is a corresponding line. Intuitively, erasure coding requires less storage space to reach the availability level than the other two with the same average peer availability.

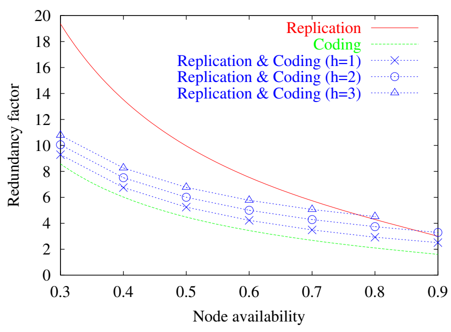


**Fig. 1.** Required redundancy factor for three nines of per-file availability, as a function of average peer availability, for the replication, coding and hybrid schemes as determined by Equation (2), (4) and (6)

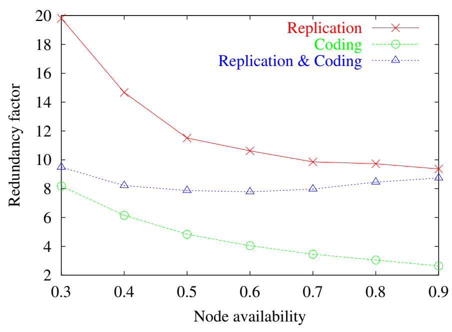


**Fig. 2.** Required redundancy factor for three nines of per-file availability, as a function of average peer availability, for the replication, coding and hybrid schemes in simulation.

The hybrid scheme's redundancy factor is slightly larger than erasure coding, and saves more storage space than replication except when average peer availability is extremely high.

Each peer periodically registers shared files and cached fragments in $M$ distributed and independent indexes. A peer locates a file with two kinds of indexes: whole file location index and fragment location index. When requesting a file $d$, a peer randomly refers to one or more indexers responsible for $d$. If the checked indexers do not provide enough whole file or fragment location information, the peer will turn to other indexers. If all $M$ distributed indexers fail to provide enough location information, the peer will wait a period of time and do the above procedure again, until the maximum lookup time is reached. If the peer can not find a whole file living in system, it turns to gather enough fragments to resemble the original file. The downloaded and resembled files are regarded as shared.

Each indexer periodically adjusts the availability of its indexed files. For file $d$ whose availability is below target level, the indexer consigns a peer holding file $d$ to increase its availability by generating and pushing necessary number of fragments to randomly selected peers. For those files without a complete copy, the adjustment will be either delayed until a user download event happen, or performed as downloading enough fragments to reconstruct original file and issue fragments by the indexer itself when maximum waiting time is reached. Here, it is not necessary to use the mechanism as erasure coding scheme to maintain a complete file in system, because almost all the files have at least one copy in the system. Such, the hybrid scheme saves the bandwidth on maintaining a copy of file on home peer.

## 3 Evaluation

We implemented the three schemes for high availability in a discrete-event packet level simulator, p2psim [8]. The simulated network consists of 1024 peers. Each peer alternately crashes and re-joins the network; the interval between successive events for each peer is exponentially distributed with a mean of given time. When a peer crashes, all files, fragments and indexes on it are discarded. Each time a peer joins, it uses a different IP address and DHT identifier. There are 1000 same sized files in the system. Distribution of requests follows Zipf-like distribution, in which relative probability of requests for the $i$'th most popular file is proportional to $1/i^{\alpha}$, where $\alpha$ is set as 0.74 (average of six traces shown in [11]). We did two sets of experiments: different peer availability and different lookup rate. In different peer availability, average peer availability ranges from 30% to 90%. In different lookup rate, average number of lookups during peer's live time ranges from 2 to 20. Each simulation runs for a simulation time of 6 hours; statistics are collected only during the second half of the simulation time. We use $m = 7$, which is the number of fragments to reconstruct original object as used in CFS [6]. The target file availability is set to 99.9% which is the availability that end users might expect from today's web services [12]. Finally, each data point in our plots represents the average over 5 trials.

We evaluate three redundancy schemes using two primary metrics:

1. *Redundancy factor* is the total storage used to achieve target availability divided by storage needed to store one copy of the whole file.
2. *Bandwidth ratio* is the total maintenance bandwidth incurred due to (1) maintaining file availability, and (2) maintaining a copy of each file on home peer for erasure coding scheme, divided by total bandwidth due to serving file requests. Bandwidth on maintaining routing table and looking up is neglectable relative to maintenance bandwidth (1) and (2). A bandwidth ratio of 0.1 implies that the bandwidth overhead of maintaining availability is 10% as much as the system must consume for normal operations.

Bandwidth ratio is regarded as more important factor in this paper, since idle bandwidth is scarcer relative to idle disk space.

## 3.1 Redundancy Factor

In Figure 2, each line corresponds to a particular scheme for high availability. Figure 2 demonstrates that the erasure coding scheme's line goes generally the same as predicted in Figure 1, but the replication scheme's does not, especially when peer availability goes beyond 60%. While the erasure coding scheme makes the least use of user downloaded files, leaving only a fragment in cache, the replication scheme shares the whole user downloaded file. Meanwhile, the higher average peer availability is, the less copy loss rate is. The replication scheme's redundancy factor remains high with high average peer availability, due to too many copies of popular files living in system.

Figure 2 also shows that although average peer availability varies from 30% to 90%, the hybrid scheme's redundancy factor changes not obviously, between 8.5 and 9.4. When peer's churn rate is intensive, the hybrid scheme takes the advantage of erasure coding to save required storage space. When peer's average availability is high, the hybrid scheme's redundancy factor does not continue falling, and even increase instead. Its reason is the same as the replication scheme: too many copies of popular files living in system. This extra redundancy is harmless. Useless copies can be discarded by user or replacement function.

## 3.2 Bandwidth Ratio

Figure 3 shows that the replication scheme saves more bandwidth than the erasure coding scheme when average peer availability is higher than 48%, and the erasure coding scheme performs better than the replication scheme in the other case. The replication scheme shares user download files to reduce the time and transfer load on maintenance. The replication scheme is effective in communities with high average peer availability, because most files are kept at desired availability level by user downloads. But in highly dynamic communities, due to frequent peer joining and leaving, user downloads do not compensate for the loss of copies. In this case, the erasure coding scheme shows its advantage in achieving higher availability increment than replication does with the same bandwidth consumption; or conversely, requiring less bandwidth for the same increment of availability level.

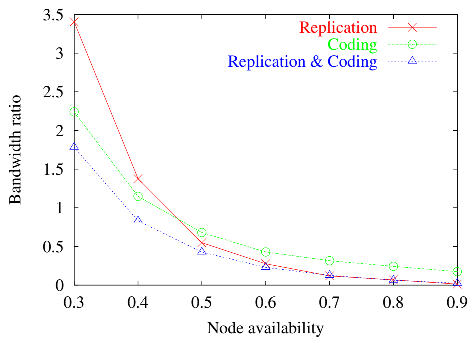
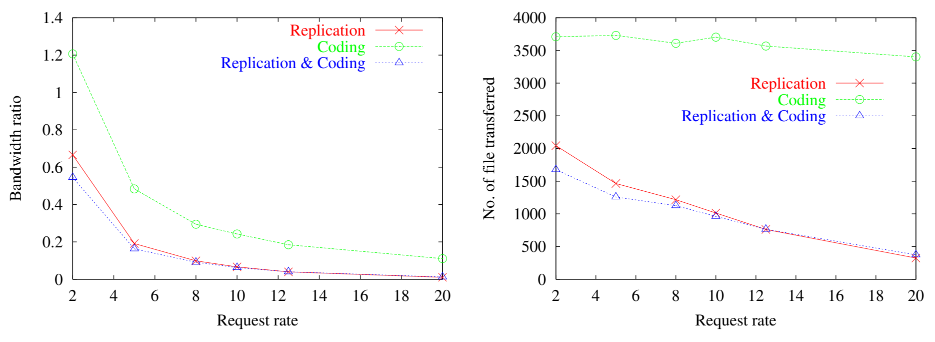


**Fig. 3.** Bandwidth ratio for three nines of per-file availability, as a function of average peer availability, for replication, coding and replication+coding

The highlight of Figure 3 is that the hybrid scheme of replication and coding achieves the best overall performance on bandwidth ratio. The hybrid scheme makes use of user download files as replication scheme, and maintains availability using erasure coding. When average peer availability is higher than 70%, the replication and the hybrid scheme consume approximately the same bandwidth on maintenance, because almost all of files' availability is high enough. When average peer availability is lower than 70%, the hybrid scheme's advantage is obvious. The hybrid scheme shares user downloaded files for subsequent accesses to save maintenance bandwidth. Another reason why the hybrid scheme saves more maintenance bandwidth than the erasure coding scheme is that the hybrid scheme do not need extra mechanism to maintain a copy of the file on home peer.

Figure 4(a) shows the situation which we might expect to see in a corporate or university environment with average peer availability is 80.7% [4]. It demonstrates that the more intensive request rate is, the less bandwidth ratio requires. While bandwidth ratio is a relative criterion, the absolute bandwidth overhead should also be paid attention to as shown in Figure 4(b). Figure 4(b) shows that while the replication and the hybrid



(a) Bandwidth ratio (b) No. of file transferred

**Fig. 4.** Bandwidth ratio and number of file transferred on maintenance for three nines of per-file availability when average peer availability is 80.7%, as a function of lookup rate, for replication, coding and replication+coding. Where request rate is average number of requests issued during a peer's lifetime.

scheme's number of transferred files on maintenance[1] falls with increment of request rate, erasure coding scheme's absolute maintenance bandwidth overhead decreases not obviously. This proves that sharing user downloaded files for subsequent accesses will considerably reduce the bandwidth on maintenance.

Figure 4 also demonstrate that the hybrid scheme of replication and erasure coding achieves better performance on bandwidth saving, especially when user request rate is low. When request rate is larger than 10, the replication and hybrid scheme's bandwidth ratio are extremely adjacent and close to x-axis. File or fragment transfer is rarely performed, because most of files' availability is maintained at desired level by abundant user downloaded files.

## 4 Conclusion and Future Work

This paper takes user download behavior into account to evaluate redundancy schemes in data storage and share systems. Experiment results show that unlike previous comparisons argued: the replication scheme saves more bandwidth than the erasure coding scheme, although it requires more storage space, when average peer availability is higher than 48%. When average peer availability is higher than 70%, the replication scheme consumes approximately the same bandwidth on maintenance as the hybrid scheme. Besides, the replication scheme introduces less complexity into system than the other two. So the replication scheme is a good choice, in high peer availability environments, e.g. university environment.

[1] Bandwidth overhead of the erasure coding scheme and the hybrid scheme is measured in terms of fragments. For comparison, their transferred number of fragments should be converted to number of files.

The erasure coding scheme requires less storage space to reach the availability level than replication with the same average peer availability, and consumes less maintenance bandwidth in highly dynamic environment. But it suffers from heavier maintenance bandwidth overhead than the replication, when average peer availability is higher than 48%, and introduces complexity into the system: not only encoding and decoding of fragments, but also entire system design complexity.

The highlight of this paper is that sharing user downloaded files for subsequent accesses (replication) and meanwhile utilizing erasure coding to maintain files' availability will achieve better performance: saving more bandwidth with acceptable redundancy factor (less than 9.4). The superiority of the hybrid scheme on saving maintenance bandwidth is obviously shown when average peer availability is lower than 70%. The experiment results also show that the hybrid scheme saves more bandwidth than the other two, when user request rate is low relative to peer churn rate. The hybrid scheme not only performs well in environments with high peer availability, but also demonstrates its advantages in highly dynamic communities. The disadvantage of the hybrid scheme is that it introduces complexity into the system.

The hybrid scheme achieve the best bandwidth saving, but in highly dynamic peer communities where average peer availability is lower than 0.5, its bandwidth ratio is still high, making the storage system suffer from poor scalability. Noting that bandwidth is scarcer relative to idle disk space, the future work should focus on saving bandwidth. Designing new coding algorithms and making further use of file copies already downloaded on network may be good for bandwidth saving. However, we leave these as issues for future work. This paper did not consider the storage limitations of the peers, we will consider it with some replacement strategies in the future.

## Acknowledgements

This work is supported by the China NSF grant, the China Jiangsu Provincial NSF grant (BK2005208), the China 973 project (2002CB312002) and TRAPOYT award of China Ministry of Education.

## References

1. R. Bhagwan, S. Savage, and G. Voelker. Replication strategies for highly available peer-to-peer storage systems. Technical Report CS2002-0726, UCSD, 2002.
2. R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. Voelker. Total recall: System support for automated availability management. In *Proc. NSDI*, 2004.
3. C. Blake and R. Rodrigues. High availability, scalable storage, dynamic peer networks: Pick two. In *Proc. HotOS*, 2003.
4. W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. In *Proc. SIGMETRICS*, 2000.
5. J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *Proc. ACM SIGCOMM*, 1998.
6. F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proc. ACM SOSP*, 2001.

7. F. Dabek, J. Li, E. Sit, J. Robertson, F. Kaashoek, and R. Morris. Designing a DHT for low latency and high throughput. In *Proc. NSDI*, 2004.
8. T. Gil, F. Kaashoek, J. Li, R. Morris, and J. Stribling. p2psim: A simulator for peer-to-peer protocols. *http://www.pdos.lcs.mit.edu/p2psim/*.
9. D. Karger, E. Lehman, F. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *Proc. STC*, 1997.
10. J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for globalscale persistent storage. In *Proc. ASPLOS*, 2000.
11. L.Breslau, P. Cao, L. Fan, G. Phillips, and S. Schenker. Web-caching and zipf-like distributions: Evidence and implications. In *Proc. IEEE INFOCOM*, 1999.
12. M. Merzbacher and D. Patterson. Measuring end-user availability on the web: Practical experience. In *Proc. IPDS*, 2002.
13. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. ACM SIGCOMM*, 2001.
14. S. Reed and G. Solomon. Polynomial codes over certain finite fields. In *J. SIAM*, 1960.
15. Rodrigo Rodrigues and Barbara Liskov. High availability in DHTs: Erasure coding vs. replication. In *Proc. IPTPS*, 2005.
16. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. Middleware*, 2001.
17. A. Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Proc. SOSP*, 2001.
18. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. ACM SIGCOMM*, 2001.
19. H. Weatherspoon and J. Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In *Proc. IPTPS*, 2002.