J Supercomput (2008) 43: 183–198
DOI 10.1007/s11227-007-0126-4

Insight into redundancy schemes in DHTs

Guihai Chen · Tongqing Qiu · Fan Wu

Published online: 31 March 2007

Abstract In order to provide high data availability in peer-to-peer (P2P) DHTs, proper data redundancy schemes are required. This paper compares two popular schemes: replication and erasure coding. Unlike previous comparison, we take user download behavior into account. Furthermore, we propose a hybrid redundancy scheme, which shares user downloaded files for subsequent accesses and utilizes erasure coding to adjust file availability. Comparison experiments of three schemes show that replication saves more bandwidth than erasure coding, although it requires more storage space, when average node availability is higher than 47%; moreover, our hybrid scheme saves more maintenance bandwidth with acceptable redundancy factor.

Keywords Peer-to-peer · Distributed hash table · Redundancy · Replication · Erasure coding

1 Introduction

The last several years have seen the emergence of a class of structured peer-to-peer systems, like CAN [1], Chord [2], Pastry [3], Tapestry [4], Viceroy [5], Cycloid [6] and so on. These P2P systems can be viewed as providing scalable, fault-tolerant distributed hash tables (DHTs). DHTs propose a determined object locating service, and already are used in many applications [7–10]. However, to provide high data

G. Chen (✉) · T. Qiu
State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing, China
e-mail: gchen@nju.edu.cn

T. Qiu
e-mail: qtq@dislab.nju.edu.cn

F. Wu
Department of Computer Science and Engineering, SUNY at Buffalo, Buffalo, USA
e-mail: fwu2@cse.buffalo.edu

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 [7, 10] and erasure coding [8, 9].

Some comparisons [8, 11, 12] 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 [13, 14] 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. Although preferring the latter one, unlike previous comparison, we take user download behavior into account and focus on both the storage and bandwidth cost in different environment. Further more, we argue that sharing user downloaded files for subsequent accesses (replication) and meanwhile utilizing erasure coding to maintain file availability will achieve better performance: saving more bandwidth with acceptable redundancy factor. There are two reasons. 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:

- 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 in structured DHTs.
- This paper demonstrates that replication saves more bandwidth than erasure coding, although it requires more storage space, when average peer availability is higher than 47%.
- 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 introduces background of replication and erasure coding. Section 3 presents related work. Section 4 describes and formulates three schemes for high availability: replication, erasure coding and the combination of them. Section 5 evaluates these three schemes by two sets of experiments. Finally, we conclude the paper and point out future work in Sect. 6.

2 Background

Redundancy schemes have been widely used in the field of distributed systems to achieve high data availability. The original form of redundancy is perhaps complete replication [10, 15]. It imposes extremely storage overhead to achieve the simplest implement of redundancy. Another common method is the parity scheme such as RAID [16]. It reduces the storage overhead but does not provide the robustness necessary to survive the high rate of failures expected in the wide area. An erasure code provides redundancy without the overhead of strict replication [17]. Erasure codes divide an object into m fragments and recode them into n fragments, where $n > m$. We call $r = n/m$ the *redundancy factor* of encoding. A rate r code increases the storage cost by a factor of r. The key property of erasure codes is that the original object can be reconstructed from any m fragments. For example, using an $r = 4$ encoding on a fragment divides the fragment into $m = 16$ fragments and encodes the original m fragments into $n = 64$ fragments, increasing the storage cost by a factor of four. Erasure codes are a superset of replicated and RAID systems. For example, a system that creates four replicas for each fragment can be described by an $(m = 1, n = 4)$ erasure code. RAID level 1, 4, and 5 can be described by an $(m = 1, n = 2)$, $(m = 4, n = 5)$, $(m = 4, n = 5)$ erasure code, respectively.

Different schemes own different characteristics. Accordingly, the choice of proper scheme depends on the specific environment and application.

3 Related work

Due to the administrative heterogeneity and poor host availability found in the P2P environment, almost all P2P systems provide some mechanism for ensuring data availability in the presence of failures.

Many systems which utilize replication scheme have been proposed. CFS [7], PAST [10] and FARSITE [15] rely on a static replication factor coupled with an active repair policy. Ranganathan et al. [18] proposed an approach in which each peer in the system possesses a model of the P2P storage system that can be used to determine number and location of replicas needed to maintain desired availability. Total Recall [11] calculates the appropriate redundancy mechanisms according to past behavior of nodes in the system.

There are two most relevant works to ours. Oceanstore [9] uses a combination of simple replication for read benefit and fragment level erasure coding for long term durability to tolerate transient failures. But it is hard to predict the durability of data in the dynamic P2P system. Another relevant work is done by Cuenca-Acuna et al. [19]. They addressed the question of increasing the availability of shared files, using an erasure coding based replication algorithm with global index, in P2P communities. However, the global index introduces single point of failure.

4 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 file availability. All of them work upon consistent hashing [20], as used by storage systems such as CFS [7].

4.1 General description

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

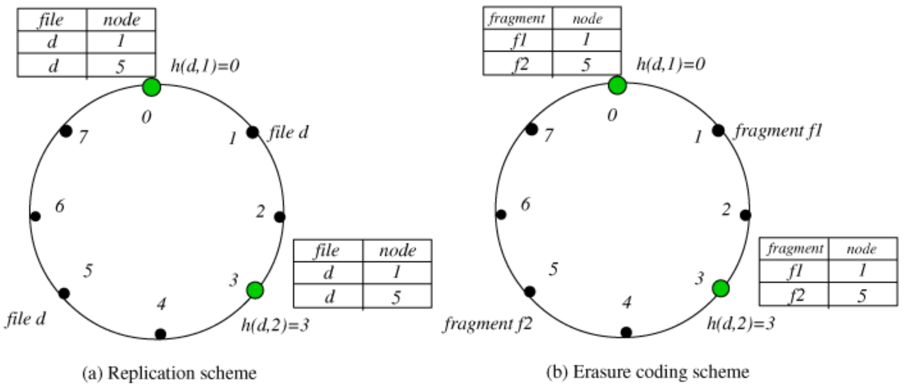


Fig. 1 An example of multiple indexes for the file d in both replication and erasure coding environment. Here, we set $M = 2$. The location of M indexes determined by $h(d, n)$ is $Node0$ and $Node3$. In figure **a**, there are two replications of file d locating at $Node1$ and $Node5$, while in figure **b** two different fragments $f1$ and $f2$ of file d locating at $Node1$ and $Node5$, respectively

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. Figure 1 gives an example of multiple indexes for file d in both replication and erasure coding environment.

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: $h(d, n)$, where $n \in [1, M]$. 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.

4.2 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.

4.3 Erasure coding scheme

Erasure codes (e.g., Reed-Solomon [17] or Tornado [22]) 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 [21]), we get the following formula for the erasure coding redundancy

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

a	σ_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

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 σ_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.

From Eq. 4 we get that redundancy factor is not relevant to n, in other words, n is not a factor influencing storage space needed to maintain a target availability level. While p is a natural property of P2P communities, m ultimately determines the redundancy level. Redundancy level falls when m increases. Larger m can reduces redundancy, but it will introduce more complexity and download delay in heterogeneous environments.

When erasure coding is used, an indexer that generates new fragments to adjust availability must have the access to the whole file. It is not scalable 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. See Fig. 2. 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.

There are two strategies to regenerate fragments:

1. The most uses of erasure codes generate all n fragments, and over time, detect and regenerate specific lost fragments. But this approach has two significant disadvantages for highly dynamic environments: (1) It is necessary to accurately account which peer is storing which fragment to regenerate lost fragments due either to peers leaving the community permanently or storage replacement; (2) It must be possible to differentiate accurately between peers temporarily going offline and

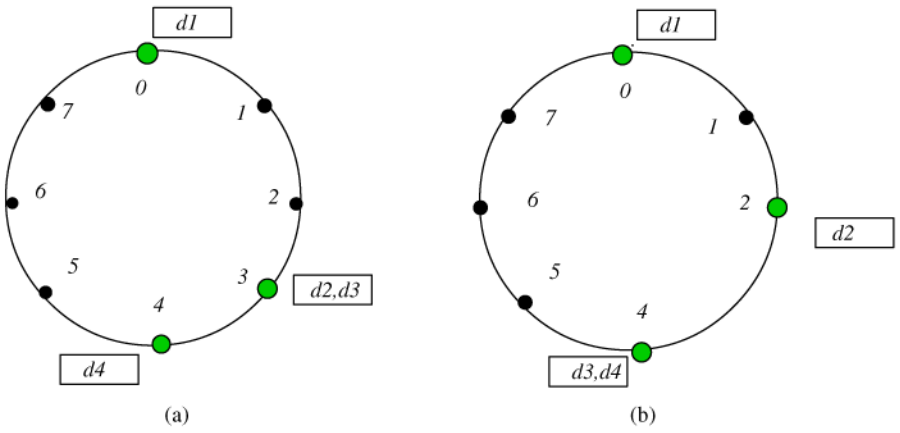


Fig. 2 An example of *home peers*. In figure **a**, $Node0$ is the home peer of $d1$. $Node3$ is the home peer of $d2$ and $d3$. $Node4$ is the home peer of $d4$. Figure **b** indicates the case of $Node3$ leave the network. $d2$ is closer to $Node2$'s identifier, so $d2$ is moved to $Node3$. $d3$ is closer to $Node4$'s identifier, so $d3$ is moved to $Node4$

leaving the community permanently to avoid introducing duplicate fragments. These two points reduce the effectiveness of erasure coding.

2. Set $n \gg m$, without generating all n fragments. When increasing a file availability, the corresponding home peer simply generates a random fragment from the set of n possible fragments. If n is large enough, usefulness of fragments will be high even if no coordination between peers, due to possibility of overlapping fragments is small. The complexity of encoding and decoding a random fragment is $\Theta(m)$ when $n - m \geq m$. So large n will not introduce large encoding or decoding complexity. To apply Eqs. 3 and 4 in our strategy, here we redefine n as the number of replicated fragments in the index.

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.

4.4 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 file 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 Eqs. 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 Eq. 5 as follows:

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

Equation 6 indicates that redundancy factor is relevant to m and h, instead of n. Redundancy level falls when m increases, with fixed h. The fragments regeneration strategies also apply to the hybrid scheme.

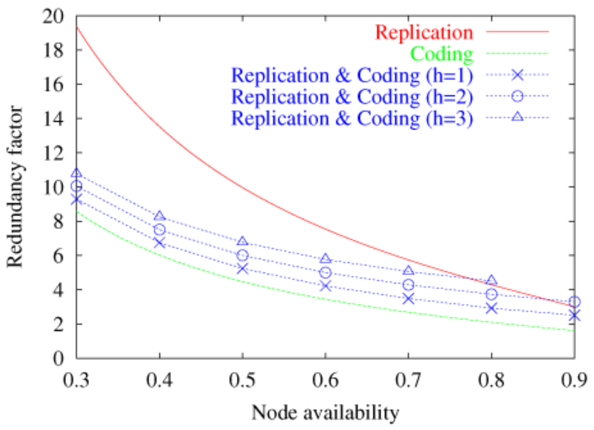


Fig. 3 Required redundancy factor for three nines of per-file availability, as a function of average peer availability, for the replication, coding and hybrid schemes as determined by Eqs. 2, 4 and 6

Figure 3 captures the theoretical redundancy factor for the replication, erasure coding and hybrid schemes determined by Eqs. 2, 4 and 6 to achieve three nines of per-file availability. Here we set $m = 7$, which is the number of fragments to reconstruct original object as used in CFS [7]. 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. 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.

Table 2 Parameter selection

Parameter description	Value/type
The number of nodes	1024
The distribution of nodes dynamic behavior	Exponential distribution
The distribution of requests	Zipf-like distribution
The parameter of the requests distribution (α)	0.74
Target file availability	99.9%
Peer availability	30%–90% (80.7% as default)
The average number of lookups per peer	2–20 (10 as default)
The number of fragments for reconstruction (m)	7

5 Evaluation

We implemented the three schemes for high availability in a discrete-event packet level simulator, p2psim [23]. 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. 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 α is set as 0.74 (average of six traces shown in [24]). 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 [7]. The target file availability is set to 99.9% which is the availability that end users might expect from today's web services [25]. Finally, each data point in our plots represents the average over 5 trials. The parameters and their values are shown in Table 2.

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.

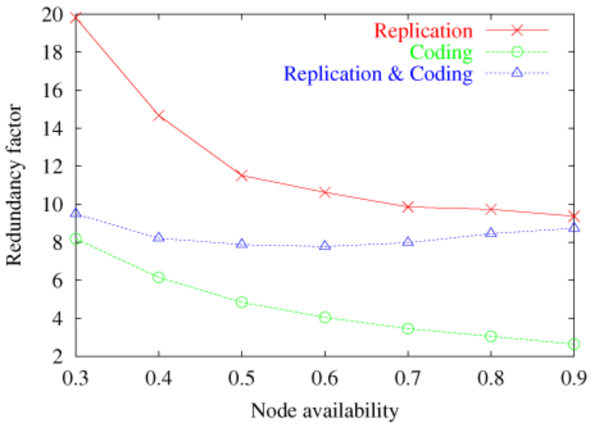


Fig. 4 Required redundancy factor for three nines of per-file availability, as a function of average peer availability, for the replication, coding and hybrid schemes in simulation

5.1 Redundancy factor

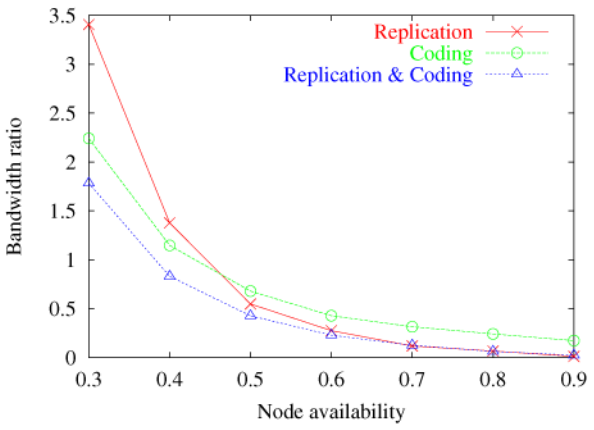
In Fig. 4, each line corresponds to a particular scheme for high availability. Figure 4 demonstrates that the erasure coding scheme's line goes generally the same as predicted in Fig. 3, but the replication scheme's does not, especially when peer availability goes beyond 60%. While the erasure coding 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 4 also shows that although average peer availability varies from 30% to 90%, the hybrid scheme's redundancy factor does not change 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.

5.2 Bandwidth ratio

Figure 5 shows that the replication scheme saves more bandwidth than the erasure coding scheme when average peer availability is higher than 47%, 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. 5 Bandwidth ratio for three nines of per-file availability, as a function of average peer availability, for replication, coding and replication + coding



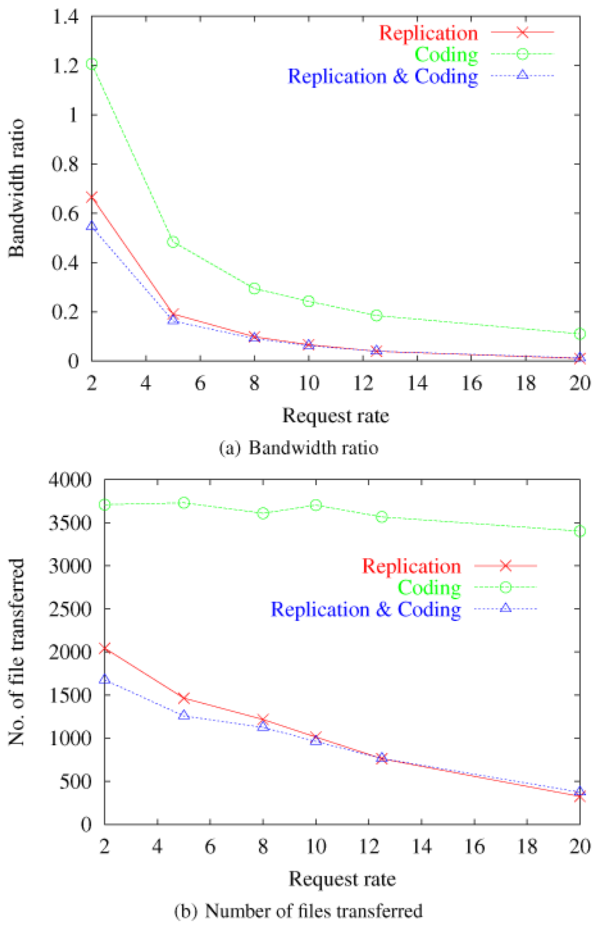
The highlight of Fig. 5 is that the hybrid scheme of replication and coding achieves the best overall performance on bandwidth ratio. The hybrid scheme makes use of user download files as replication scheme, and maintains availability using erasure coding. When average peer availability is higher than 70%, the replication and the hybrid scheme consume approximately the same bandwidth on maintenance, because almost all of file 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 6 shows the situation which we might expect to see in a corporation or university environment with average peer availability is 80.7% [15]. 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 Fig. 6b. Figure 6b shows that while the replication and the hybrid 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 6 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 file availability is maintained at desired level by abundant user downloaded files.

[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.

Fig. 6 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



6 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 47%. 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.

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 47%, 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 file 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.

Acknowledgement This work is supported by the China NSF grants (60573131, 60673154), the China Jiangsu Provincial NSF grant (BK2005208), the China 973 project (2006CB303004) and TRAPOYT award of China Ministry of Education.

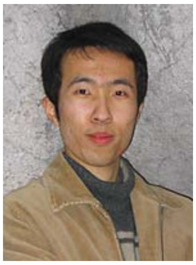
References

1. Ratnasamy S, Francis P, Handley M, Karp R, Shenker S (2001) a scalabe content addressable network. In: Proc of ACM SIGCOMM, 2001, pp 161–172
2. Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H (2001) Chord: a scalable peer-to-peer lookup service for Internet applications. In: Proc of ACM SIGCOMM, 2001, pp 149–160
3. Rowstron A, Druschel P (2001) Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer system. In: Proc of Middleware, 2001, pp 329–350
4. Zhao B, Huang L, Stribling J, Rhea SC, Joseph AD, Kubiatowicz JD (2004) Tapestry: a resilient global-scale overlay for service deployment. IEEE Trans Select Areas Commun 22(1):41–53
5. Malkhi D, Naor M, Ratajczak D (2002) Viceroy: a scalable and dynamic emulation of the buttterfly. In: Proc of Principles of Distributed Computing, 2002, pp 183–192
6. Shen H, Xu CZ, Ghen G (2004) Cycloid a constant-degree and lookup-efficient P2P overlay network. In: Proc of IPDPS, 2004, pp 26–30
7. Dabek F, Kaashoek MF, Karger D, Morris R, Stoica I (2001) Wide-area cooperative storage with CFS. In: Proc of ACM SOSP, 2001, pp 202–215
8. Dabek F, Li J, Sit E, Robertson J, Kaashoek F, Morris R (2004) Designing a DHT for low latency and high throughput. In: Proc of NSDI, 2004, pp 85–98
9. Kubiatowicz J, Bindel D, Chen Y, Czerwinski S, Eaton P, Geels D, Gummadi R, Rhea S, Weatherspoon H, Weimer W, Wells C, Zhao B (2000) Oceanstore: an architecture for global-scale persistent storage. In: Proc of ASPLOS, 2000, pp 190–201
10. Rowstron A, Druschel P (2001) Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In: Proc of SOSP, 2001, pp 188–201
11. Bhagwan R, Tati K, Cheng Y, Savage S, Voelker G (2004) Total recall: system support for automated avalability management. In: Proc of NSDI, 2004, pp 337–350
12. Weatherspoon H, Kubiatowicz J (2002) Erasure coding vs. replication: a quantitative comparison. In: Proc of IPTPS, 2002, pp 328–338
13. Blake C, Rodrigues R (2003) High availability, scalable storage, dynamic peer networks: pick two. In: Proc of HotOS-IX, 2003, pp 1–6
14. Rodrigues R, Liskow B (2005) High availability in DHTs: erasure coding vs. replication. In: Proc of IPTPS, 2005, pp 226–239

15. Bolosky WJ, Douceur JR, Ely D, Theimer M (2000) Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. In: Proc of SIGMETRICS, 2000, pp 34–43
16. Patterson D, Gibson G, Katz R (1988) The case of raid Redundant arrays of inexpensive disks. In: Proc of SIGMOD, 1988, pp 109–116
17. Reed S, Solomon G (1960) Polynomial codes over certain finite fields. J SIAM 8:300–304
18. Ranganathan K, Iamnitchi A, Foster I (2002) Improving data availability through dynamic model-driven replicatiion in large peer-to-peer communities. In: Proc of CCGRID, 2002, p 376
19. Cuenca-Acuna FM, Martin RP, Nguyen TD (2003) Autonomous replication for high availability in unstructured P2P systems. In: Proc of SRDS, 2003, pp 99–108
20. Karger D, Lehman E, Leighton F, Levine M, Lewin D, Panigrahy R (1997) Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on world wide web. In: Proc of STC, 1997, pp 654–663
21. Bhagwan R, Savage S, Voelker G (2002) Replication strategies for highly available peer-to-peer storage systems. UCSD Technical Report CS2002-0726
22. Byers JW, Luby M, Mitzenmacher M, Rege A (1998) a digital fountain approach to reliable distribtuion of bulk data. In: Proc of SIGCOMM, 1998, pp 56–67
23. Gil T, Kaashoek F, Li J, Morris R, Stribling J p2psim: a simulator for peer-to-peer protocols. http://www.pdos.lcs.mit.edu/p2psim/
24. Breslau L, Cao P, Fan L, Phillips G, Schenker S (1999) Web-caching and zipf-like distribution: evidence and implications. In: Proc of INFOCOM, 1999, pp 126–134
25. Merzbacher M, Patterson D (2002) Measuring end-user availability on the web: practical experience. In: Proc of IPDS, 2002, pp 473–477

Guihai Chen obtained BS degree from Nanjing University, M. Engineering from Southeast University, and PhD from University of Hong Kong. He visited Kyushu Institute of Technology, Japan in 1998 as a research fellow, and University of Queensland, Australia in 2000 as a visiting professor. During September 2001 to August 2003, he was a visiting professor in Wayne State University. He is now a full professor and deputy chair of the Department of Computer Science, Nanjing University. Prof. Chen has published more than 100 papers in peer-reviewed journals and refereed conference proceedings in the areas of wireless sensor networks, high-performance computer architecture, peer-to-peer computing and performance evaluation. He has also served on technical program committees of numerous international conferences. He is a member of the IEEE Computer Society.

Tongqing Qiu received his BS degree in computer science from Nanjing University, China, in 2004. He served as a research assistant in the Department of Computer Science at City University of Hong Kong from April 2006 to February 2007. He is now a master-degree student in the Department of Computer Science at Nanjing University, China. His research interests are in the areas of peer-to-peer computing and distributed systems.



Fan Wu acquired BS degree in computer science from Department of Computer Science, Nanjing University, China, in 2004. Now he is a PhD candidate of Department of Computer Science and Engineering, the State University of New York at Buffalo, U.S.A. His research focuses on incentives and privacy in wireless networks, privacy preserving data mining, and peer-to-peer computing.