On the Impact of the Data Redundancy Strategy on the Recoverability of Friend-to-Friend Backup Systems

Marcelo Iury S. Oliveira, Walfredo Cirne, Francisco Brasileiro, Dalton Guerrero

Universidade Federal de Campina Grande
Departamento de Sistemas e Computação
Laboratório de Sistemas Distribuídos
Av. Aprígio Veloso, s/n, Bodocongó
58.109-970, Campina Grande, PB, Brasil
{iury,walfredo,fubica,dalton}@dsc.ufcg.edu.br

Abstract. *Social network-based systems, also known as Friend-to-Friend (F2F) systems, are a promising approach to develop backup solutions that provide high reliability with a much lower consumption of bandwidth and storage than P2P ones. F2F backup systems can use two data redundancy strategies to handle peer failure events, namely: replication and erasure coding. In this paper we evaluate the use of these alternatives to handle failures in F2F backup systems. The assessment is conducted using a new metric named recoverability slowdown. The proposed metric represents how efficiently one can restore the data lost due to the occurrence of failures. Our aim is to determine which data redundancy scheme constitutes a more balanced solution in terms of recoverability slowdown under different network bandwidth capacities and storage overhead levels. The simulations we have performed indicate that an increase on the storage overhead leads to better values for the backup recoverability slowdown when using the replication technique, while it leads to worse values when using the erasure coding technique. Moreover, using enough redundancy, replication can achieve results that are close to the optimal case for recoverability slowdown, while the best performance achieved by erasure coding is not larger than 87% of the optimal case. Nevertheless, for relatively low values of storage overhead, erasure coding outperforms replication.*

1. Introduction

Performing backup of data is relevant to all computer users. Nevertheless, only a small portion of them conduct this management activity in an appropriate way. While hardware failures make it unwise to use free disk space to backup files stored in the same device, excess disk space can be used to backup files from other computers. Furthermore, the capacities of modern hard disks have outgrown the needs of most home users, leaving them with plenty of idle storage space [Douceur and Bolosky 1999]. These facts suggest the organization of collections of PCs into a distributed and collaborative system providing a mutual backup service. In fact, some P2P systems based on this idea have been recently proposed [Landres et al. 2004, Cox et al. 2002, Batten et al. 2001].

These systems have aimed at building reliable storage from unreliable components. Inevitably, backup copies can be compromised by permanent failures suffered by these components. Reliability is a measure on the likelihood of data being permanently lost or corrupted, formally defined as the probability of the survival of at least one copy of data for a specified period of time *t*. This definition makes a clear distinction between permanent failures (when the data is definitely lost) and transient ones (when the data remains inaccessible for a limited period of time). Thus, to be able to control reliability, a system needs to be concerned about all the failures that may permanently compromise the access to its functions. P2P storage systems replicate data to provide high reliability in the presence of failures. This is done in such a way that every time a copy is lost, new copies of the data are made on other peers. It is no surprise that performing such copies, however, may consume large amounts of bandwidth, turning reliability an expensive commodity [Blake and Rodrigues 2003]. Anonymous P2P systems are highly vulnerable to the participation of free riders, particularly if they do not implement an incentive mechanism to promote cooperation [Dingledine et al. 2003, Acquisti et al. 2003]. Free riders can liberate storage space simply by removing backup data of other peers stored in their computers, compromising the backup reliability. Finally, anonymous P2P systems or systems in which new identities can be cheaply obtained are susceptible to white-wash and Sybil attacks [Pontes et al. 2007]. As a consequence, distributed backup systems in which peers are anonymous must assume that peers are not trustworthy and, therefore, require an increased storage overhead to deal with free riding. This also implies that more bandwidth must be used to create new copies of data, turning reliability even more expensive.

By using a social network to form a P2P system instead of anonymous peers, interactions with possibly malicious peers are prevented or at least minimized [Sabater and Sierra 2002, Marti et al. 2004, Staab et al. 2005, Pouwelse et al. 2006]. Thus, a friend-to-friend (F2F) backup system will have a low incidence of free riding, allowing it to provide high reliability with a much lower consumption of bandwidth and storage. Li and Dabek (2006) have derived an analytical model for F2F systems. They evaluated the model and showed that this assumption drastically reduces the maintenance bandwidth required.

F2F backup systems can use two data redundancy techniques to handle peer failures: replication or erasure coding. Comparing how erasure coding can provide a more robust solution in terms of efficient usage of network resources in file systems than replication is a topic that has been widely researched. Despite the abundant literature, no definitive choice has been universally agreed upon, and no easy rule of thumb has been devised to guide system designers. While these researches have focused on ensuring data availability and reliability, we propose the use of recoverability slowdown as another performance metric to analyze and compare data redundancy techniques and their efficiency in F2F backup systems.

Recoverability slowdown represents how efficiently one can restore the data lost due to the occurrence of failures, while reliability is concerned with whether the backup is recoverable or not, recoverability expresses how quickly a backup can be recovered. The ability to recover quickly from a system failure or disaster depends not only on

having backup copies created of one's data, but also on having an efficient system that appropriately recovers data on the highly dynamic environment of F2F backup systems.

Thus, our objective is to determine if a F2F backup system can truly meet the time frames expectations for data recovery. In particular, the recoverability analysis uses a failure model that is appropriate to represent the failures to which a F2F backup system is subject. We simulate different scenarios, using statistical distributions observed from traces extracted from live P2P systems. Diversity and dynamics of peers of real P2P are also considered in the simulations. Finally, we discuss several aspects of a F2F backup service based on social networks, called OurBackup.

The rest of the paper is organized as follows. In Section 2 we provide introductory information concerning social networks and present some benefits of social networks in a backup system. In Section 3 we give a brief overview of the two data redundancy techniques evaluated. The recoverability slowdown metric is presented in Section 4, while in Section 5 we describe its use in evaluating F2F backup systems. In Section 6 we discuss some related work, while a brief overview of the OurBackup system is presented in Section 7. Finally, in Section 8 we present our conclusions and indicate the future steps of this work.

2. The Role of Social Networks

Social networks are representations of existing relationships between people or organizations [Barnes 1972]. People become related in different levels, both qualitatively and culturally. These relationships are associated to social rules, which portray the way each human being (or organization) is inserted in a community. The communities also have a set of rules, which are collaboratively determined by their members. Thus, the social networks are systems capable of congregating individuals and institutions in a decentralized way.

With the advent of the Internet, tools that offer new forms of relationship, communication and organization have emerged. Instant messaging systems (*e.g.* Yahoo, MSN, ICQ) monitor the users' friends to inform their connectivity status (*e.g.* on-line, off-line). Relationship sites, like Orkut, Friendster, Facebook and LinkedIn, allow the construction of new relationships and communities. These tools have allowed relationships in the real world to have representations in the virtual world, under the same social rules.

Social networks are valuable in the development of computational systems because they capture reliable relations between entities of the real world. These relations may be used to improve the efficiency and trustworthiness of reputation mechanisms [Sabater and Sierra 2002], routing and placement solutions [Marti et al. 2004] and file sharing systems [Staab et al. 2005].

In a P2P backup system, recovery time depends on the available bandwidth, on the size of the backup set, and on the time peers stay on-line while the recovery is performed. Here, again, the social network shows its benefits. We can expect that the probability of a friend being connected to the system is influenced by the social network. Users stay on-line because they want to help their friends as much as they want to be helped by them. Wright and Ayton (2005) have observed that one's experiences of regret impact on one's subsequent thinking and behavior. They showed

that students when imagine themselves in a scenario in which they need to regularly make backup, they choose to improve their backup process to avoid a negative outcome. Hence, we can accept that friends are more likely to feel regret if they remove their friends' backup than they would in relation to anonymous peers', therefore decreasing the probability of a backup to be corrupted due to free riding events. Moreover, friends can be contacted using out-of-band communication channels such as phone, email, or instant messaging systems and asked to connect to the system when help is required. Furthermore, if a friend goes off-line, it is also possible to use out-of-band information to determine whether the nature of the failure is permanent or transient. In a sense, we can consider the users of a F2F system to be part of it, in the so-called *social layer*.

3. Data Redundancy Techniques

In F2F backup systems, it is a challenge to satisfy reliability and recoverability requirements due to the autonomic and unexpected behavior of peers. Usually, peers are under control of individual users who turn their machines on and off at their discretion. In spite of the existence of such independent and unexpected behaviors, data redundancy techniques allow F2F backup systems to ensure reliability and recoverability, even if participant peers are not connected all the time and may experience permanent faults. Two techniques of data redundancy typically used on storage systems are replication and erasure coding.

Replication is the simplest redundancy technique. Using replication, a system stores k exact copies of the data on distinct nodes. The erasure coding technique provides redundancy without the overhead of strict replication. Erasure coding divides a file into b blocks and recode them into $k \times b$ blocks with overlapping redundancy, which are then distributed to distinct nodes. The key property of erasure coding is that the original file can be reconstructed from any b blocks out of the $k \times b$ recoded blocks.

In both cases, the cost of the F2F backup system is determined by the storage overhead (k). When replication is used, k is the number of copies of the data. For erasure coding, k is the expansion factor (*i.e.* amount of redundancy added to the file) used in the codification of the blocks. Erasure coding also defines b as the amount of blocks generated in the coding of the file. The values of k and b are defined based on the reliability and recoverability requirements specified and on the maximum permanent failure rate expected for the system. Their values are chosen a priori, *i.e.*, before failures occur. Thus, as we do not know which copies will be compromised due to failures, all we can do is to assume that the copies and the blocks fail permanently with probability f.

The choice between the two techniques has been the subject of numerous studies and publications [Lin et al. 2004, Weatherspoon and Kubiatowicz 2002, Rodrigues and Liskov 2005]. Lin et al. (2004) have analyzed and compared the data availability of these two data redundancy approaches performing a series of experiments to measure how erasure coding and replication respond to changes in peer availability. They have also studied the storage overhead these redundancy schemes impose. They concluded that erasure coding works better than replication when the peer availability is high, and vice versa. Their availability formulae can be easily derived to evaluate reliability requirements. It is necessary only to switch the peer availability for the peer failure probability.

While research on data redundancy techniques has focused on data availability and reliability, we focus on recoverability, instead. In the following sections, we analyze how replication and erasure coding affect recoverability requirements for F2F backup systems.

4. Recoverability Slowdown

The recoverability attribute more referenced in literature is the Mean Time to Repair (MTTR) [Kraus 1988]. MTTR is the expected time to recover a system from a failure. Similar to MTBF (Mean Time Between Failures), MTTR is typically expressed in hours. While MTTR impacts availability, it does not impact reliability. In a F2F backup system, MTTR is the recovery time of a backup, which is the time to perform the download of the entire backup. The best possible (optimum) recovery time is given by S/d, where d is the download capacity of the peer that is recovering the backup and S is the size of the backup to be recovered.

Note that an optimum recovery time is only achieved when there is backup data available that can be downloaded at the maximum rate d. In a F2F system, this might not always be the case. The number of peers that are on-line, as well as their upload capacity, ultimately defines the rate with which the requesting peer recovers its backup.

To gauge the recoverability of a system in a way that is independent on the capacity of the recovering peer, we define a new metric, named the recoverability slowdown (R), $0 < R \leq 1$. R is defined as the relation between the optimum recovery time and the actual recovery time (RT) of a backup:

$$R = \frac{\left(\frac{S}{d}\right)}{RT} = \frac{S}{RT \times d} \quad (1)$$

Hence, R expresses how close or far from the optimum a given system is. The closer R is to 1 the better the system.

5. Recoverability Analysis

The backup recoverability is affected by a large spectrum of factors. For example, whenever a peer becomes off-line, the data it stores becomes inaccessible, temporarily disabling the recovery of backups and, as a consequence, increasing RT. It is difficult to model each possible event that can impact the backup download process. Moreover, these events greatly vary with regards to their values. Studies carried out by Saroiu et al. indicated that the set of peers participant of the systems Napster and Gnutella is heterogeneous with respect to many characteristics: capacity bandwidth, latencies, lifetime, shared data etc. [Saroiu et al. 2002]. In fact, values vary between three and five orders of magnitude among peers.

Simulation normally allows modeling a system much closer to reality than an analytical model, increasing the significance of the results. Scenarios of simulation in which we vary different characteristics of peers (*e.g.* upload and download capacities) and their backup datasets (*e.g.* file sizes) allow us to verify the influence of each parameter in the backup recoverability. We, therefore, opted to study the backup

recoverability slowdown metric by considering dynamic environments in a simplified model and simulating this model using information from real traces extracted from live P2P systems.

5.1. Simulation Model

We have developed a discrete event simulator, on which the typical entities of the backup recovery process are modeled. In each simulation run, a peer is either a consumer or a provider. A peer acts as a *consumer* when it submits backup recovery requests to others peers. It acts as a *provider* when it receives and takes care of recovery requests. We believe that the probability of two or more consumers belonging to the same social network recover a backup simultaneously is negligible small. Thus, for the sake of simplicity, we assume that there is a single consumer and that all other peers are providers, collectively storing the backup set of the consumer peer.

The behavior of providers is determined by the states it can assume: UP (*i.e.* the provider is online), DOWN (*i.e.* the provider is offline) and FAILED (*i.e.* the provider failed permanently; this can happen because of a disk crash, for instance). The consumer peer can only submit backup recovery requests to the providers which are in the UP state. When there is more than one provider in the UP state, the consumer adopts a greedy approach, requesting data transfers from all online providers.

All providers store the same amount of data; however, the stored content varies depending on the data redundancy technique applied. When replication is used, all peers store the same content, which is a complete backup of the dataset. When erasure coding is used, peers store different data sets, because the applied codification process results in the creation of blocks with distinct contents. In this case, each peer stores one codified block of each file.

We consider the file concept in the simulator to verify the impact of the differences in content between peers. Instead of having a single block of data that represents, for example, a backup of 10GB, we have a set of several files that together sum 10GB of data. The file size distribution was based on studies performed by Crovella et al. (1998). They estimated the distribution of file sizes of the UNIX file system as a Pareto distribution with parameters α=1.05 and β=3,800.

The availability pattern of the peers was based on empirical observations performed by Stutzbach and Rejaie (2006). Their observations suggest that the Weibull distribution (shape, scale) provides a good model for the length of peer sessions, representing a compromise between the exponential and Pareto distributions. Based on their empirical data we used a Weibull distribution with *shape*=0.59 and *scale*=40 to generate the join and leave events that will lead peers to assume the UP and DOWN states, respectively.

Another factor that affects the result of the simulation is the peer bandwidth. Our values are based on the service provided in Brazil. All the providers and consumers use the same bandwidth capacity. This simplification was based on data from the Comitê Gestor da Internet no Brasil (CETIC), which indicates that most of Internet users have the same capacity [CETIC 2008]. We believe this represents an approximation of the average case of the real world. Moreover, varying the bandwidth capacity would not

result in significant changes, since we used a normalized metric that reduces the impact in the measurements of the bandwidth actually used.

As the ADSL (Asymmetric Digital Subscriber Line) technology is so widespread, we assume that upload capacity (u) differs from the download capacity (d). Moreover, when the sum of the providers' upload bandwidth overreaches the consumer's download bandwidth, a bandwidth partitioning procedure is applied. A smart download bandwidth division can improve the recoverability of the backup, when the sum of the upload bandwidth of all on-line peers is larger than the download bandwidth of the requesting peer. Instead of performing an even division of the bandwidth, we prioritized peers that remain connected to the system for longer periods. This is done by aggregating the remaining session time of all providers to calculate the percentage of each provider in relation to the whole session time. This percentage is then used to decide the size of download portions that the providers will receive. The session time of each provider can be gathered by tracking the peers' join and leave history. In the simulator this is provided by an oracle which has knowledge about the overall system. Thus, the bandwidth division is proportional to the session time of each peer.

The execution of a simulation scenario is composed of the workload, a set of providers that store backups and a consumer interested in the recovery of the backup. The workload is defined by three parameters: dataset size to be recovered (S), storage overhead (k) and erasure coding fragmentation (b). Values of k and b vary from 1 to 10 units. The amount of providers is defined by k and $k \times b$ for replication and erasure coding, respectively. Note that keeping the same number of providers for the erasure coding case, by allocating several blocks/copies of the same file in a provider, reduces the backup reliability, since the failure of one provider will corrupt $k > 1$ blocks/copies.

5.2. Failure Model

F2F backup systems are subject to occurrence of failures. According to studies of the Ontrack Data Recovery [OnTrack 2007], a North American company specialized in data protection, the main causes of data loss are: disk failures (56%), human error (26%), software bugs (9%), viruses (4%) and disasters (2%). Thus, we can expect that failures in a social network-based P2P backup system are dominated by disk failures.

One of the most common models to represent the time to fail is the exponential distribution [Kraus 1988]. The function that describes the probability of occurrence of disk failures for a finite interval of time Δt is given by [Kraus 1988]:

$$f(\Delta t) = 1 - e^{-(\Delta t / MTBF)}, \quad (2)$$

where Δt indicates the time of interest (*e.g.* the necessary time to recover the backup), and MTBF (Mean Time Between Failures) represents the mean time between disk failures. The disk failure time used in our simulations was generated through the exponential distribution with the parameter 1/MTBF. We considered, in our analysis, an MTBF of 300,000 hours that represents a pessimistic hypothesis of MTBF [Maxtor 2007].

This function represents a random variable that indicates the probability of failure of the providers' disk during backup recovery. When a disk failure occurs in a

provider, we assume that stored backups are corrupt and should therefore be ignored by the consumer in the recovery process. Since the simulations aimed to establish the backup recovery time, once a consumer has completely recovered the backup, the simulation process is finalized. Therefore, it may be possible that this happens before a disk failure occurs. Nevertheless, we considered the occurrence of disk failure events because we wanted to verify the influence of these events on the recoverability of F2F backup systems.

5.3. Results

For each scenario simulated we have performed enough executions to produce results with a 95% confidence level and an error smaller than 5%. Figure 1 shows the results obtained by simulating six different scenarios using replication as the redundancy technique. These scenarios represent different backup dataset sizes and bandwidth capacities. A surprising result is that scenarios with d = 300 Kbps and u = 150 Kbps provided better recoverability slowdown than scenarios with d = 1 Mbps and u = 300 Kbps. This happens because for larger bandwidth values, there is a higher probability of not enough providers to be available to entirely allocate the download channel of the requesting peer. If we increase the consumers' download capacity without a proportional increase on the providers' upload capacity, the probability that a consumer does not use completely its download capacity raises and, consequently, the recoverability is reduced.

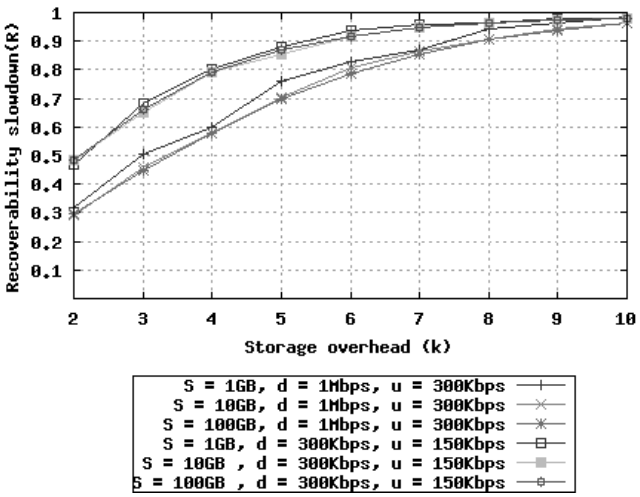

Figure 1. Backup recoverability slowdown using replication

An expected result we can also observe analyzing Figure 1 is that backup recoverability slowdown improves as storage overhead increases. When using replication, all providers store the same content, allowing the consumer to recover backup from any set of peers and also resuming download processes interrupted from any peer. Amongst the presented scenarios, we can observe that the recoverability slowdown gets very close to R = 1 for k = 10, meaning that a practical backup system can get close to the optimum recovery time even in a highly dynamic environment.

However, one must remember that a better recoverability slowdown does not necessarily imply in a smaller *RT*. Figure 2 shows the *RT* gotten through simulation of the same scenarios presented in Figure 1. Analyzing these results, one can observe two

obvious behaviors. First, the *RT* improves as the peers' bandwidths increase; this happens because the amount of data that peers can transfer increases. Secondly, the *RT* increases linearly in respect to the backup dataset size.

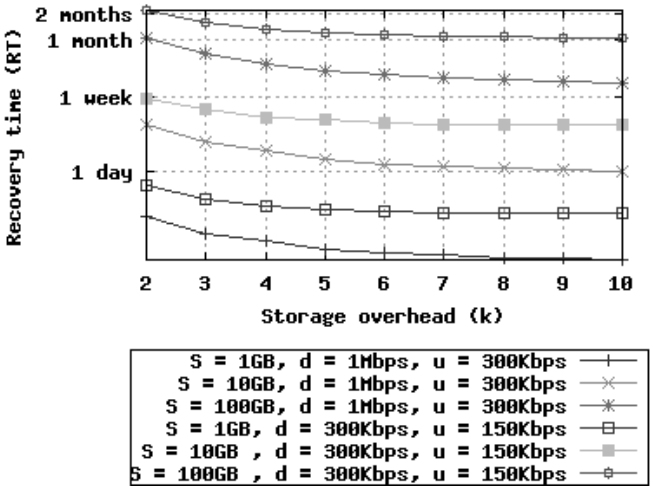

Figure 2. Recovery time using replication

Regarding erasure coding, increasing the *k* and *b* values does not necessarily result in a recoverability improvement. Different from replication, when using erasure coding it is necessary to recover at least *b* complete blocks to completely recover a file. As each peer stores different data content, it is not possible to restore the data transference from another peer. Moreover, the amount of providers that can participate of the download process increases, because there are $k \times b$ peers storing the backup data. This raises the probability that the sum of providers' upload bandwidth overreaches the consumer's download bandwidth and, as a consequence, increases the time needed to download a complete block.

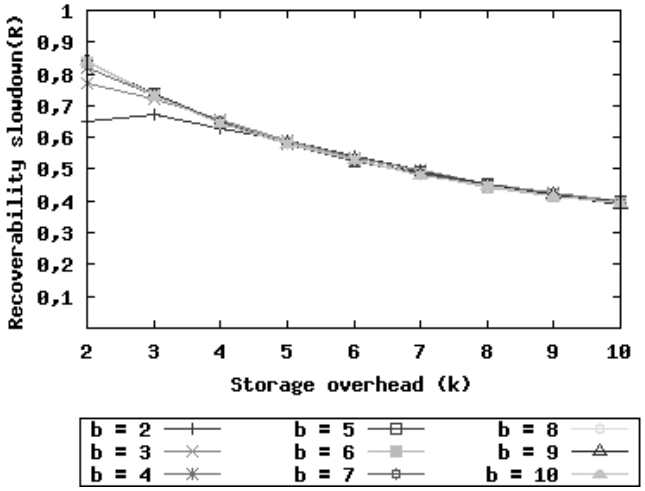

Figure 3. Backup recoverability slowdown using erasure coding

Figure 3 shows the results obtained by simulating ten different scenarios using erasure coding. Analyzing these results, we can observe that the backup recoverability slowdown decreases as storage overhead increases. Again, this happens because, when the storage overhead increases, the probability that the consumers' download bandwidth

is surpassed by the sum of the providers' upload bandwidth also increases. Consequently, smaller bandwidth slices are allocated to each provider, while the total amount of data that needs to be recovered keeps constant. The smaller the portion of the download bandwidth allocated, the longer is the time necessary to recover a block and the higher is the probability of a provider becoming off-line, interrupting the transference process. Since blocks have distinct contents, it is not possible to re-establish interrupted transferences from a different provider; consequently the recovery of a file only happens when b blocks are entirely recovered. Naturally, once b blocks have been fetched, all the other unfinished transferences related to this file are discarded, decreasing the consumer aggregated transference rate.

Furthermore, recoverability slowdown keeps relatively steady for most cases as block fragmentation increases. Similarly to *k*, when the value of *b* increases, the amount of providers also increases, raising the probability of the sum of the providers' upload bandwidth overreaches the consumer's download bandwidth. However, the amount of data that needs to be fetched from each provider (*S/b*) decreases, because when there is an increase on the block fragmentation, there is also a decrease on the size of the blocks to be codified.

Finally, analyzing Figure 4, we can compare the recoverability slowdown of replication and erasure coding, obtained through simulation. We perceive that for $k < 5$, erasure coding offers better recoverability than replication. This happens because smaller values of *k* reduce the probability of having providers on-line, increasing thus the necessary time to recover a backup. However, for $k > 5$, the replication technique surpasses erasure coding, regarding the recoverability slowdown metric.

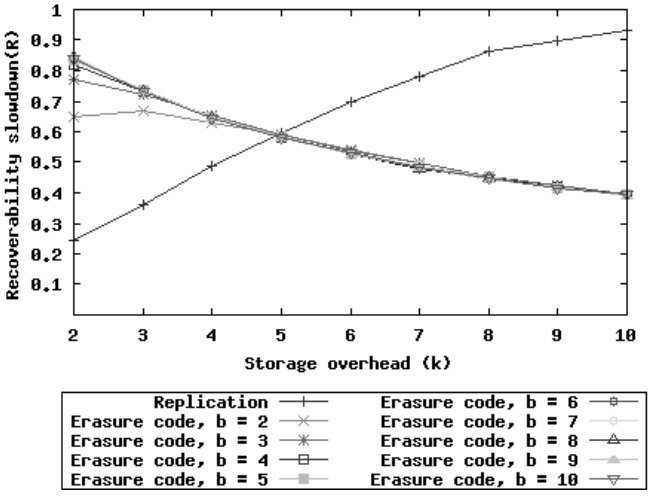

Figure 4. Replication versus erasure coding

5.4. Discussion

The analysis of data redundancy techniques to support F2F backup systems allows us to provide a better understanding of when erasure coding or replication outperforms one another in respect of recoverability slowdown. Surprisingly, our results indicate that, for the case of erasure coding, an increase on the storage overhead leads to worse values for the backup recoverability slowdown. Moreover, provided that enough redundancy is used, the recoverability slowdown achieved by replication is close to 1, the optimal

value, while the best performance achieved by erasure coding is not larger than 0.87. Nevertheless, for small values of storage overhead ($k < 5$) erasure coding still offers better recoverability than replication.

However, it is important to remember that to provide storage overhead greater than one, the nodes of collaborative backup system must donate an amount of resources that is directly proportional to k. Therefore, the use of erasure coding is presented as an alternative to increase the recoverability slowdown without increasing the consumption of resources.

6. Related Work

There are several proposals of P2P systems [Landers et al. 2004, Cox et al. 2002, Batten et al. 2001] that exploit unused storage space to backup data. Most of them aim to build a large-scale, reliable and available backup system from many small-scale unreliable, lowly-available distributed hosts.

As mentioned before, these related systems assume anonymous peers, making it difficult to differentiate between permanent and temporary departures. Therefore, they have to consider all departures as permanent failures, resulting in higher peer failure rates. As a consequence, maintaining high reliability becomes as costly as maintaining high availability, and therefore, like P2P storage, P2P backup systems are doomed not to work [Blake and Rodrigues 2003].

Li and Dabek (2006) have demonstrated the feasibility of using social networks in P2P systems. However, their work does not present any study about the recovery time of backups or which data redundancy technique is better suited for social network-based P2P backup systems. Backup systems have their performance measured in terms of the backup time as well as the recovery time. In this work we performed a simulation analysis to assess the performance and the feasibility of F2F backups systems using different data redundancy techniques.

Comparisons of the erasure coding and replication techniques have been the subject of several studies reported in the literature. Rodrigues and Liskov (2005) focused on comparing erasure coding and replication concerning DHTs, and they concluded that when peer availability is high, replication is preferred, whereas in scenarios where peer availability is low erasure coding is the preferred scheme. Weatherspoon and Kubiatowicz (2002) quantitatively compared these data redundancy techniques in terms of bandwidth usage and storage overhead. They concluded that erasure coding uses an order of magnitude less bandwidth and storage than replication. Lin et al. (2004) conducted a series of experiments to measure how erasure coding and replication respond to changes in peer availability, as well as the storage overhead each of these redundancy schemes imposes. However, none of them focused on comparing data redundancy schemes in terms of recoverability.

7. The OurBackup Solution

We explored the idea of a F2F backup system with the construction of OurBackup. OurBackup is a P2P backup system based on social networks. It allows a user to use friends' free storage space to backup her own files and to donate space to let her friends backup their files on her PC. Users register themselves in OurBackup and take the

responsibility of constructing their own social network. This process is not different from that used by widely known social network systems like Orkut, Friendster, Facebook and LinkedIn. A search engine helps the user finding and inviting friends. As soon as invitations are accepted, the OurBackup agent is able to use friends' donated space to backup the user's files and vice versa. OurBackup uses a public key cryptographic system and hash checksums to encrypt and secure all backup data. Hence, although data is copied to friends' computers, no friend is able to read or modify it.

The OurBackup architecture consists of a central server and agents. The server is responsible for managing users and their social networks providing the authentication services and the maintenance of security copies of all the vital information of the system (*e.g.* backup metadata and users' social networks). Agents are responsible for the execution of backups, and are partially independent from the server: each agent operates with metadata stored at a local catalog, and periodically synchronizes with the server. Users can interact with OurBackup from agents installed at any machine. If the user is not at her machine, the agent fetches metadata directly from the server. This allows a user to recover backups in case of machine crashes. Backup copies are transferred directly between agents.

We designed OurBackup to support both replication and erasure coding. The major difference in OurBackup design is the unit of data sent out to peers during back up and retrieval: replication uses copies and erasure coding uses blocks. Moreover, erasure coding needs decomposition and reconstruction methods for files to go from file to block and vice versa. Because of its greater simplicity, we initially implemented the OurBackup using replication, however, in a future version we will also assess the introduction of erasure coding.

8. Conclusions

In this work, we proposed a new recoverability metric to evaluate backup systems. We used this metric to analyze the impact of using two different data redundancy techniques, namely erasure coding and replication, in a social network-based backup system. Our analysis considered a set of relevant parameters for backup systems, such as storage overhead levels, bandwidth capacities, peer dynamics behaviors and node failure probability.

The results we presented showed that increasing the storage overhead does not always imply in an improvement on the recoverability slowdown achieved. In particular, when the replication technique is used, the expected improvement in the recoverability slowdown is obtained as the storage overhead increases, while in the case of erasure coding the behavior is the opposite. Nevertheless, replication provides worse backup recoverability slowdown than erasure coding for relatively low values of the storage overhead ($k < 5$).

Another surprising result was that when upload capacity differs from the download capacity, increasing the bandwidth will not result on a linear improvement of the recoverability slowdown. This behavior happens because not always the consumer peer obtains enough peers to completely fulfill its download capacity.

There are several interesting issues left for further studies. The analysis in this paper assumed that all the peers have the same failure probability. The study of backup

recoverability when peers have different failure probability is an interesting direction. As pointed out by many others, the failure events may be correlated to each other [Bhagwan et al. 2003].

Acknowledgments

This work has been partially developed in collaboration with HP Brazil R&D. We are indebted to Alexandro Soares, Eduardo Colaço, Milena Michelli, Nazareno Andrade and Paolo Soares who helped us in the development of OurBackup. Francisco Brasileiro thanks the support received from CNPq/Brazil (grant 309033/2007-1).

References

Alessandro Acquisti, Roger Dingledine, and Paul Syverson. (2003) On the Economics of Anonymity. In Rebecca N. Wright, editor, Proceedings of Financial Cryptography (FC ’03). Springer-Verlag, LNCS 2742.

John A. Barnes. (1972) Social Networks. Addison-Wesley. Module in Anthropology.

Christopher Batten, Kenneth Barr, Arvind Saraf, and Stanley Trepetin. (2001) pStore: A secure peer-to-peer backup system. Technical Memo MIT-LCS-TM-632, MIT Laboratory for Computer Science.

Ranjita Bhagwan, Stefan Savage, and Geoffrey Voelker. (2003) Understanding availability. In Proc. 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03).

Charles Blake and Rodrigo Rodrigues. (2003) High availability, scalable storage, dynamic peer networks: Pick two. In Ninth Workshop on Hot Topics in Operating Systems.

CETIC. (2008) Velocidade da conexão à internet utilizada no domicílio. http://www.cetic.br/usuarios/tic/2007/rel-geral-05.htm, Accessed in March 2008.

Landon P. Cox, Christopher D. Murray, and Brian D. Noble. (2002) Pastiche: making backup cheap and easy. SIGOPS Oper. Syst. Rev., 36(SI):285–298.

Mark E. Crovella, Murad S. Taqqu, and Azer Bestavros. (1998) Heavy-tailed probability distributions in the world wide web. pages 3–25.

Roger Dingledine, Nick Mathewson, and Paul Syverson. (2003) Reputation in P2P Anonymity Systems. In Proceedings of Workshop on Economics of Peer-to-Peer Systems.

John R. Douceur and William J. Bolosky. (1999) A Large-Scale Study of File-System Contents. In Proceedings of the 1999 ACM SIGMETRICS, pages 59-69, Atlanta, Georgia.

J. W. Kraus. (1988) Maintainability and reliability. In: IRESON, W. G.; COOMBS JR. Handbook of reliability engineering and management. New York: McGraw-Hill, cap. 15, p.15.1-15.38.

Martin Landers, Han Zhang, and Kian-Lee Tan. (2004) PeerStore: Better performance by relaxing in peer-to-peer backup. In P2P’04: Proceedings of the Fourth International Conference on Peer-to-Peer Computing (P2P’04), Washington.

Jinyang Li and Frank Dabek. (2006) F2F: reliable storage in open networks. In Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS'06).

W. K. Lin, D. M. Chiu, and Y. B. Lee. (2004) Erasure code replication revisited. In P2P '04: Proceedings of the Fourth International Conference on Peer-to-Peer Computing (P2P'04), USA. IEEE Computer Society.

Sergio Marti, Prasanna Ganesan, and Hector Garcia-Molina. (2004) Sprout: P2P Routing with social networks. In volume 3268 of Lecture Notes in Computer Science.

Maxtor. (2007) Maxtor diamondmax16 datasheets. WebPage, Accessed in: January, 2007.

OnTrack. (2007) Understanding data loss. http://www.ontrackdatarecovery.com.au/understandingdataloss/ , Accessed in: January, 2007.

Felipe Pontes, Francisco Brasileiro, Nazareno Andrade. (2007) Sobre Calotes e Múltiplas Personalidades no BitTorrent. Anais do III Workshop de Redes Peer-to-Peer. Belém do Pará. p. 75-86.

J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J.Yang, A.Iosup, D.Epema, M.Reinders, M. van Steen, H.Sips. (2006) Tribler: A Social-Based Peer-to-Peer System. In Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS'06).

R. Rodrigues and B. Liskov. (2005) High availability in DHTs: Erasure coding vs. replication. In Proc. of the IPTPS.

Jordi Sabater and Carles Sierra. (2002) Reputation and social network analysis in multi-agent systems. In AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems.

Stefan Saroiu, P. Krishna Gummadi, and Steven Gribble. (2002) A measurement study of peer-to-peer file sharing systems. In SPIE Multimedia Computing and Networking (MMCN2002).

Steffen Staab, Pedro Domingos, Peter Mika, Jennifer Golbeck, Li Ding, Tim Finin, Anupam Joshi, Andrzej Nowak, and Robin R. Vallacher. (2005) Social networks applied. IEEE Intelligent Systems, 20(1):80–93.

Daniel Stutzbach and Reza Rejaie. (2006) Understanding churn in peer-to-peer networks. In IMC '06: Proceedings of the 6th ACM SIGCOMM on Inter-net measurement, pages 189–202, New York, NY, USA. ACM Press.

H. Weatherspoon and J.D. Kubiatowicz: (2002) Erasure coding vs. replication: A quantitative comparison. In: Proc. 1st International Workshop on Peer-to-Peer Systems (IPTPS'02), Cambridge, Massachusetts, United States.

C. Wright and P. Ayton. (2005) Focusing on what might happen and how it could feel: can the anticipation of regret change students' computing-related choices?. Int. J. Hum.-Comput. Stud. 62(6), 759-783.