Optimizing peer-to-peer backup using lifetime estimations

Samuel Bernard, Fabrice Le Fessant

Optimizing Peer-to-Peer Backup using Lifetime Estimations

Samuel Bernard
samuel.bernard@lip6.fr

INRIA Saclay/UPMC LIP6, France

Fabrice Le Fessant
fabrice.le_fessant@inria.fr

INRIA Saclay – Île de France, France

ABSTRACT

In this paper, we study the viability of a peer-to-peer backup system on nowadays internet connections. In particular, we show that peer lifetime estimation can be used to reduce the maintenance cost of peer-to-peer backup. Previous studies [5] have shown that lifetimes in a peer-to-peer system follow a Pareto distribution. Consequently, peers can be sorted on their expected lifetimes, depending only on the length of their history in the system. By carefully selecting the peers on which backup data is stored, repairing cost can be highly reduced for long-term backup users, while it is still acceptable for new users. The efficiency of this technique is evaluated through simulations of a state-of-the-art peer-to-peer backup system.

1. INTRODUCTION

Backup of data has never been as important as in today's life. The presence of computers (digital cameras, organizers, etc.) in every part of our lives has considerably increased the quantity of personal digital data. In the same way, companies are now highly dependent on vital information stored on their computers. As a consequence, efficient techniques to prevent the loss of these data have to be used on every computer.

Yet, few computers are correctly configured to protect their data. Traditional backups on external supports are very common, but also unexpectedly very inefficient: 20% of nightly backups on tapes fail, while 40% are unusable afterwards; systems are not well configured, or just badly maintained. Few people know that backups on CD-R are short term (two years). Finally, all these solutions are very slow, and require additional hardware or manual interventions, and do not protect against thieves, fires or floods. Service Storage Providers (SSP) have received a lot of attention lately, with the cloud-computing wave, but still suffer from other risks, such as bankrupcies, internal failures (in Hotmail and Gmail) or lack of security.

Therefore, our long term objective is to provide a new alternative backup system, free, efficient, reliable, yet still easy-to-use, to help people protect data on their personal computers. Our system would be based on the exchange of free disk space between its members. To lower the cost and to bring a high resilience to faults, it has to be decentralized and to work in a peer-to-peer (P2P) way. In the literature, there are already some P2P backup systems. But most of them are still in prototype phase and has not yet been tested in acceptable conditions. The other ones are too restrictive and do not offer the conditions we want. Before building a complete system, we want to simulate such systems in order to know what their viability is, based on common properties of already designed systems.

Our contributions. In this paper we present the general description of a backup system and the specifications needed to estimate its viability. Then we describe our simulations and analyse their results. The contributions are twice. We prove the viability of a general peer-to-peer backup system and we introduce a new criteria, the age, to estimate the reliability of a peer.

2. SPECIFICATIONS

In this section, we introduce the general model and the basic assumptions for this work. We also describe the behavior of a state-of-the-art peer-to-peer backup system.

2.1 Model

We consider a large-scale network (in the order of thousands of computers) composed of nodes (or peers), connected by an underlying communication medium, typically IP. In this network, we assume full connectivity between peers, i.e. any peer can communicate with any other peer. This can usually be done by relaying the communications for firewalled peers, and we will not discuss it further.

Peers in the network have some disk space, divided between data that they need to backup and free disk space that they can share with other peers.

Studies of real peer-to-peer file-sharing networks [5, 16, 23] have shown that the lifetime of peers in these systems is highly correlated to the length of their presence in the system. In other words, the longer a peer has been in the system, the longer it is expected to stay in the system. We assume that this property will also be observed in peer-to-peer

backup systems, especially as peers are likely to participate more if the safety of their data is at stake.

To take this into account in our system, we assume the existence of a secure monitoring protocol for peer availability[17, 14]: any peer can query the availability of any other peer for a given period of time, for example the last 90 days.

We assume that peers have enough computation resources to run all the parts of the backup system. Indeed, this paper focuses on optimizing the bandwidth used by the backup system, and not on optimizing the computation power, as the bandwidth is often the most limitating resource for home users. Consequently, we do not focus on the confidentiality of data: standard cryptography can be used to ensure data confidentiality, for example by encrypting data before it is used by the backup system.

Finally, we assume that the backup system uses erasure codes, such as Reed-Solomon[19], instead of simple replication to increase the redundancy of the backup. With such codes, for each set of k blocks of data, the system generates m blocks of correcting codes. These $n = k + m$ blocks are then stored in the system. To get back the data, any k blocks from the n final blocks can be used. As an example, if $k = 128$ and $m = 128$, the system will store the data on 256 different nodes using twice the initial storage, but supporting until 128 node failures without losing any data. With replication, using twice the storage would mean that the data is only replicated twice, and so data might be lost after only two failures.

2.2 Description of the Backup System

A peer-to-peer backup system can be divided into three main tasks:

Backup of data : how the data is moved from the computer to the network

Restoration of data : how the data is moved from the network back to the computer

Maintenance of data : how the data is preserved in the network

In the following, we describe these three tasks in more details. Some of these details might differ in different systems, but this description can easily be adapted to most peer-to-peer backup systems.

2.2.1 Backup of data

During the backup task, new data (either the content of complete files or the diffs between versions) is collected on the file-system, and is stored in a single file (archive). A new archive is created when the previous one reaches a given size. Usually, meta-data is stored in a different archive, with a better redundancy, to speed up the restoration task. For confidentiality, data in each archive can be encrypted using a session key.

Each archive is splitted in k blocks, and $n = k + m$ new blocks are created. With Reed-Solomon, the k first blocks are the original ones, but these blocks might be different from the original ones in other coding systems. The n blocks are then uploaded to n different peers in the system. If the backup system contains a direct exchange mechanism, these n partners will be allowed to store one or more blocks of data on the peer in exchange for the space they have provided. Some systems might prefer a more global policy of fairness, where space is exchanged globally (see [7] for example) instead of between partners.

Finally, a master block is created. It contains the list of peers on which data has been stored, the list of archives, in particular the ones containing meta-data, and session keys, encrypted with the user public key (and consequently, only accessible using the user personal private key). The master block is then uploaded to the network, for example to all the partners storing the peer's data or to a DHT.

2.2.2 Restoration of data

The restoration task is done in the exact opposite order of the backup task. The master block is first retrieved from the network, for example using a flooding request or a query to a DHT. Meta-data archives are then downloaded to build an index of all the files stored in the backup. These two steps are only needed if the local copy of this information stored on the user computer has been lost. The data archives are then downloaded to restore the files on the computer, using the deciphered session keys to decrypt the files if needed.

To download an archive, the peer must reach at least k of its partners for that archive. Once k blocks have been downloaded, the k original blocks are decoded from these k blocks, and the content of the archive becomes available.

2.2.3 Maintenance of data

Churn is a well known problem in peer-to-peer systems: peers participating in the network are not connected all the time, and can decide to leave the network definitively, for example if they are not satisfied by the quality of service and they rather use another software.

As a consequence of churn, all blocks are not always available, and they might completely disappear when a peer decides to leave the network. In backup systems, contrary to distributed file-systems[4], availability is less important than durability: if there is a trade-off between speed and security, the users are likely to prefer security, i.e. a better guarrantee to be able to retrieve their files, even if it takes more time. Therefore, in this paper, we are not interested in the instantaneous availability of the blocks, but only in their durability.

The maintenance of the backup is the perpetual task of replacing the blocks which have disappeared from the network. Since a peer cannot know if another peer that it cannot connect, will come back in the network or not, a time threshold is used: if a peer could not be connected during the threshold period, it is considered that the peer has definitively left the system and the blocks it was supposed to store have disappeared.

A *repair threshold* k' is usually defined, to decide when a repair operation should be triggered. The repair threshold

specifies the minimal number of blocks that should be visible in the system. If fewer blocks are visible in the system, more failures could prevent a peer from recovering its data and thus its backup would be useless.

In this paper, we assume that, to replace d blocks that have disappeared from the network for a given archive (when $n-d<k'$), the peer must first download k blocks to be able to decode the original data. It can then re-encode either the missing blocks, or new blocks. This is a worst-case choice, as alternative coding systems[8, 9] have been proposed to reduce the number of blocks needed to replace missing blocks. Finally, the peer must upload the new blocks to new partners (or to current partners but for other archives) and update the corresponding meta-data.

2.2.4 Evaluation of Maintenance Cost

From the previous description, we can evaluate the cost of maintenance of one archive with the following formula:

$$\Delta_{download} + \Delta_{decoding} + \Delta_{encoding} + \Delta_{upload} + \Delta_{metadata}$$

With recent computers, computation time for encoding and decoding is negligible compared to transfers on the network. In particular, DSL connections are usually asymmetric, with a good download bandwidth and a limited upload bandwidth. Finally, updating the meta-data is fast, as the peer just needs to upload the descriptors for the new partners, if any. Consequently, the cost of repair can be simplified to the download of k blocks and the download of d blocks:

$$\Delta_{repair} = \Delta_{download} + \Delta_{upload}$$

We set the following parameters for our backup system:

Parameter	Value
Archive Size	128 MB
k (initial blocks)	128
m (added blocks)	128

If we estimate the bandwidth of a DSL connection to 32 kB/s for upload, and 256 kB/s for download, we obtain $\Delta_{download} > 512s$ and $\Delta_{upload} > d \times 32$. Consequently, with $d < 128$, a total repair time should last $69+8 = 77$ minutes, most of which is taken by the upload of regenerated blocks. It means that no more than 20 repair operations should be triggered per day. Since the network bandwidth is shared with other applications, and a user might want to backup many archives (one gigabyte of data is 8 archives of 128 MB), the system becomes really usable on a daily basis only if the number of repairs for one archive is much lower. For example, if we want to limit the cost to one repair per day, with 32 archives (4 GB of data), the repair rate should be less than one per month approximatively. Of course, this is a worst-case example, modern DSL connections (in France) are at least four times faster, and FTTH connections are even faster.

In this paper, we show that it is possible to decrease this maintenance cost for daily users of such a system, by better choosing on which peers the data is stored. Our choice heuristics uses the age of peers to estimate their stability. In the following, we describe the simulations we made to evaluate the efficiency of our scheme.

3. SIMULATIONS

3.1 Simulation Context

To prove that an availability-aware scheme is efficient for peer-to-peer backup, we simulated a simple protocol using the PeerSim[12] simulator, a peer-to-peer simulator written and extensible in Java.

Our simulations are round-based: in a round, each peer is given the opportunity to execute some code and interact with some other peers; execution is sequential, so the code for one peer is always executed before or after the code for another peer, never concurrently, but the order of peers is chosen randomly at each round.

In our simulations, each round represents one hour. Indeed, we are not interested in small events, such as the messages which are sent or short-term disconnections. Instead, we focus on longer events, such as long-term peer disconnections and repair operations. One hour is indeed close to the approximation we did in the previous section of the time needed for a long repair (128 blocks in 77 minutes). Still, it is important to notice that repair operations do not need to be limited to one hour, since the critical part is the download of n blocks, taking usually around 8 minutes when done aggressively, while the upload of generated blocks can be done later as new partners become available. Finally, such a long duration per round allows us to simulate the system over a longer period, such as a few years, which is quite important for a long-term backup system.

To simulate a peer-to-peer backup system, we first made some assumptions:

Independence All departures, arrivals, disconnections and reconnections of nodes are strictly independent. This property was verified experimentally[2] on real peer-to-peer systems.

Fidelity A node is less likely to quit the system when it has been in the system for a long time [5, 16, 23]. This is the key property as it gives us a good criteria for the selection of partners. A node is more likely to exchange blocks of data with nodes which are as stable or more stable than it is.

Moreover we do not consider the existence of free-riders in this work, for the sake of simplicity. Yet, a peer does not rely completely on the system. It chooses his partners, evaluates them and is responsible for the maintenance of its backup. This design allows peers to completely monitor their backup and to choose different strategies independently, such as different parameters for repair thresholds. Moreover, peers only rely on the system for storing their blocks, and not to do complex tasks, such as repairs, which is safer approach. If a peer deletes some data, or is disconnected for a long time, it should still be able to recover its data. The time needed for recovery depends of the repairs rate, and the results are presented in section 4.

3.2 Simulated Protocol

In this subsection, we describe the protocol that we implemented in our simulator, in particular, during a repair operation. As written in section 2.2, it can be seen as a simple extension of existing protocols, such as [15]. We consider only one data archive per peer, the results can be trivially extended to the case of multiple archives.

When a node wants to store blocks on the peer-to-peer network, it creates a pool of possible partners, i.e. peers that do not yet store blocks for the same archive. To enter this pool, both peers must agree on their partnership, using an acceptation function that will be specified later. Once the pool is big enough, the peer can choose the d partners it needs to store the new blocks, and upload the data. Nodes are selected according to their stability. Because this stability cannot be guessed, the protocol uses the ages of the peers in the system to sort them: the longer a node has been in the system, the more stable it will be considered.

The acceptation function is used by peer p_1 to choose whether a partnership can be started with peer p_2. In our simulations, we took the following acceptation function:

$$f(p_1, p_2) = \min\left(\frac{L - (\min(s_1, L) - \min(s_2, L)) + 1}{L}, 1\right)$$

In this function, s_1 and s_2 are the estimations of the stability of both peers, i.e. the number of rounds since each peer first connected to the system. If some peers have an age greater than L, only the L part will be considered. We chose 90 days for L in our simulations, as we think it is a long enough period to consider a peer stable, and peers which have been in the system for longer times are not much different.

The result of the acceptation function is the probability for peer p_1 to accept p_2 as partners. We can notice the following properties:

The result is never zero, and actually, its minimum is $\frac{1}{L}$. Indeed, we want that the probability to be accepted as a partner is never nul, even for newcomers in the system.

The result is always one if peer p_2 is older than peer p_1. Indeed, peers should always accept older peers as partners.

The function is not symmetric: the probability for p_1 to accept p_2 is different from the probability for p_2 to accept p_1, unless both peers are older than L.

Then during each round, every peer monitors its partners, i.e. checks whether they are online and have its data (see [18] for proofs of storage). If the number of partners for an archive is below a threshold, the peer will trigger a repair to replace the missing partners.

In our simulation, the initial step, when initial blocks are uploaded for the first time to the network, is seen as a repair where $d = 256$. A peer is not considered as included in the network until that first operation finishes.

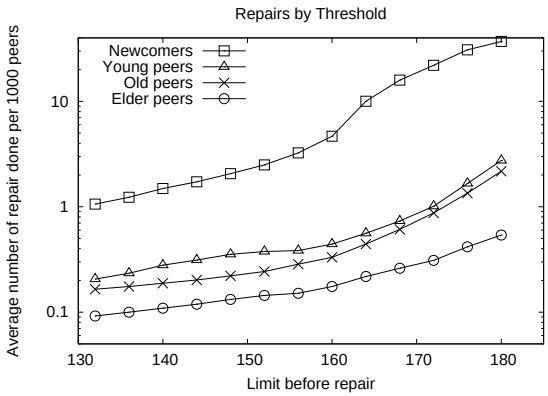

Figure 1: Average rate of repairs for the four categories of peers depending of the repair threshold.

3.3 Profiles of Peers

All the peers do not have the same behaviour in our system. Some peers are very stable and do not disconnect very often. Other peers are unavailable most of the time and are highly likely to quit the system early, consequently removing the blocks of their partners from the network.

In order to simulate this heterogeneity of behaviors, we have classified the peers into different profiles. A profile is a class of peers sharing globally the same behavior. For instance, in our simulation, there is a profile for very stable nodes, another one for very unstable nodes, etc.

In this paper, we are interested in making the system inexpensive for peers in the stable profiles, while making it a little more expensive for newcomers, who have to prove their stability to really benefit from such a system.

4. EVALUATION AND RESULTS

We first specify the parameters used in our simulations, and then, we present and discuss the results.

4.1 Parameters

In our simulations, we have used the following parameters:

- After a short growing phase, the number of peers in the system is 25,000. Each peer leaving the system is immediatly replaced, and its blocks are immediatly removed from its partners. We plan to investigate a more realistic approach in future work, where data would be removed only after a *grace period* ranging between one and several weeks.
- We only consider one archive per peer, with $k = 128$ original blocks and $m = 128$ redundancy blocks, thus $n = 256$ total blocks. However, we claim that these results should scale linearly when the number of archives of a peer is increasing, since they can be handled independently.
- We make the threshold before repair vary between 132 and 180, focusing later on the value 148.

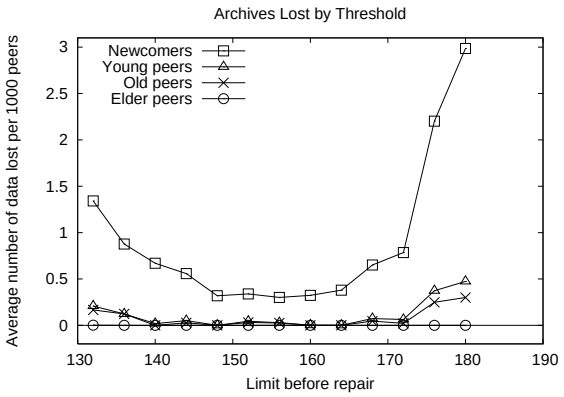

Figure 2: Average rate of data lost for the four categories of peers depending of the repair threshold

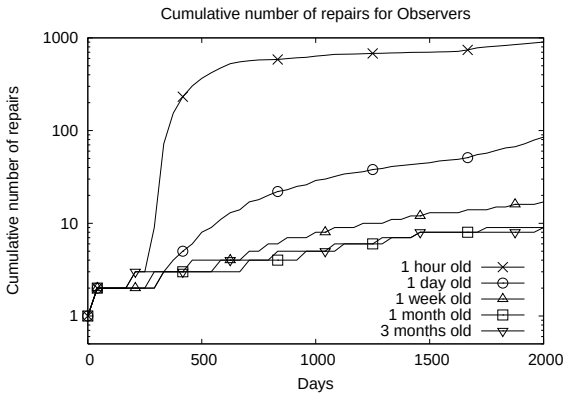

Figure 3: Total number of repairs done by observers

- A peer provides storage for at most 384 blocks in total to its partners: quota = 384. It means that a peer should provide three folds of free storage what it plans to backup in the system. We plan to investigate smaller quota in future work.
- We simulate 50,000 rounds which is approximately 5 years and half.

Note that, although our simulations are limited to 25,000 peers and one archive, results should the same for bigger systems and more archives, as peers are likely to aggregate into clusters or cliques, where all peers are partners for the other peers of the cluster, and pools for an archive are filled by partners for other archives.

4.1.1 Profiles

Each peer belongs to a profile and it cannot change during the simulation. A peer cannot know to which profile an other peer belongs to. A profile specifies different properties for a peer. Currently, there are two properties, the peer's life expectancy, which is how many rounds it will stay in the system, and its availability, which is the percentage of time it will be online. We use four different profiles, in different proportions in the system for our simulations. They are described in the following table:

Profile	Proportion	Life expectancy	Availability
Durable	10%	unlimited	95%
Stable	25%	1.5 - 3.5 years	87%
Unstable	30%	3 - 18 months	75%
Erratic	35%	1 - 3 months	33%

In short, around a third of the peers are stable and should suit to the backup, another third are less stable and could cause some problems and the last third is too erratic to be really usable. The question we ask is if the unstable peers can prevent stable ones from achieving good backup performances (in number of repairs).

We claim that this setting is a realistic one. To the best of our knowledge, no peer-to-peer backup system is largely deployed on the Internet, so no traces of user's behaviors are available. Therefore, to create the previous profiles, we used the result of studies on peer-to-peer file-sharing systems [2]. Those result are too pessimistic to create a reliable backup system, so we modified them a little to be more optimistic. However, we kept a large part of unstable peers in order to still keep a pessimistic approach. We claim that this approach is the most realistic one, as users in peer-to-peer file-sharing networks have an incentive to disconnect early (not to be caught by the police), whereas users in peer-to-peer backup networks have an incentive to remain connected (to protect their data).

4.2 Results and Discussion

In the following, we present and discuss our two main results: the impact of the repair threshold, and the cost of maintenance for the four profiles of users.

4.2.1 Impact of the Repair Threshold

Our goal was to evaluate the impact of the repair threshold on the repair rate and on the data loss rate. So we ran a set of simulations with exactly the same parameters, presented at the beginning of this section, while varying the repair threshold between 132 and 180.

We present the results for 4 categories of nodes, differentiated by their age. Note that during the life of a peer, its category changes depending on its age, whereas its profile does not change: at the beginning it is a Newcomers, then after 3 months it will be counted as a Young peer, etc. The following table details the interval of ages for each category:

Peers	Age
Elder peers	$>$ 18 months
Old peers	6 - 18 months
Young peers	3 - 6 months
Newcomers	$<$ 3 months

Figure 1 plots the average rate of repairs, for the four categories depending on the repair threshold. Figure 2 plots the average rate of data loss for the four categories, also depending on the repair threshold.

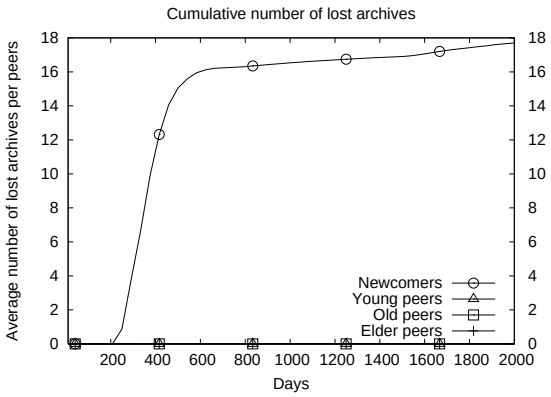

Figure 4: Evolution of the cumulative number of lost archives for the four categories of peers

Average Rate of Repairs. As expected, the main result of figure 1 is that the number of repairs increases accordingly to the repair threshold. The bigger the threshold, the more peers need to repair. The reason is that peers are progressively losing their partners and, if the repair threshold is high, it is reached sooner. The increase is not linear and raises faster after 156. Another result is the stratification between the profiles. Young peers (erratic ones) repair more often than the elder ones (stable ones). We detail this in the section 4.2.2.

Average Rate of Data Lost. As the repair threshold has an impact on the number of repairs, it has also an impact on the number of lost archives. As seen in the figure 2, if the repair threshold is too small, a peer may lose too quickly its partners, and will be unable to regenerate original blocks to fulfill the repair. For instance, if the threshold is 132 and a peer has 133 blocks available, it may lose more than 5 blocks in a round if its partners are not very stable, and consequently, with fewer than 128 blocks available in the system, be unable to repair. Even if the disconnections were temporary, the peer might never be able to repair early enough before other definitive disconnections. On the contrary, if the threshold is too high, a young peer will lose its blocks too quickly and thus will have to repair at each round. But that peer might have some difficulties to find new partners, and so the peer might be unable to upload all the missing blocks, and so, to prevent the number of its partners from decreasing dangerously.

To decide on a good repair threshold, we have to find a good compromise between the loss rate and the repair rate. As the repair rate is strictly increasing, we can take the smallest value of threshold with a good loss rate. 148 seems such a good compromise. In the following, we present in more details the simulations computed with a repair threshold set to 148.

4.2.2 Impact of the Age of Peers

In the previous results, we have seen that the stable profiles will repair less often than the unstable ones. We now explain what is happening on an example, the simulation computed with a repair threshold set to 148.

Repair Rate for Observers. Figure 3 shows the total number of repairs done by some particular peers, called *observers*. An observer is a special peer, whose age does not increase like the age of other peers. Other peers cannot choose an observer as a partner, but the observer can choose other peers as partners, without however consuming their quota (the quota is the number of blocks a peer can host for its partners, and is set to 384 blocks in our simulations). As normal peers, it has to repair if its number of available blocks decreases below the repair threshold.

In this simulation, we put 5 observers:

Observer	Age
Elder	3 months = the age limit
Senior	1 month
Adult	1 week
Teenager	1 day
Baby	1 hour

The figure shows that the repair rate strongly depends on the age of a peer. The Elder and Senior observers have less than 10 repairs in 2000 days, the Adult has less than 20 repairs, the Teenager has less than 100 repairs and finally the Baby has a huge 900 repairs. Note that the global performance are better than we may think. The Baby observer has 900 repairs but a normal peer, during its lifetime in the system, will only be considered as a Baby for one day and as a Teenager for one week. So, quickly, most of the peers will be as good as an Adult observer, or even better.

Analysis. The link between the age and the rate of repair is clearly understandable. An elder peer is accepted by every other peer, so he can choose to store its blocks on the most stable peers. Moreover, after some repairs, he has replaced most of the unstable partners that he was forced to use when he was a newcomer. Moreover, as a side-effect of the protocol, stable peers will progressively form clusters of partners.

Nevertheless, younger peers still have good performances, since usually, one week is enough to find good stable partners.

Losses of Archives. To understand why data is lost, we plotted in figure 4 the evolution of the cumulative average number of lost archives for the four categories of peers presented in the table of the section 4.2.1 (and not observers this time).

The figure shows that an Newcomers will lose about 18 archives during 2000 days in the system, while all the other peers almost never lose anything. The loss of data for Newcomer peers seems huge but we can also notice a faster increase of data loss between the 200^{th} and the 600^{th} day. This corresponds to a stabilisation phase due to the start

of the simulation where all peers have the same age. The more eloquent part of the curve is between the 1000^{th} and the 2000^{th} day when the system has became stable: we see that the total number of lost archives drop to 2 in 1000 days which is reasonable. Moreover in 1000 days, a peer would be quickly considered as at least an unstable or a stable peer and so would not suffer archives lost anymore in fact.

With these simulations, we have shown that it is possible to optimize the maintenance cost of a peer-to-peer backup system, by moving the load of maintenance from stable peers (actually, long-term users of the system) to unstable peers. Moreover, we have shown that the cost is higher for newcomers only for a short period.

5. RELATED WORK

As far as we know, no previous study of viability of peer-to-peer backup systems has been done. However, some work has already been done on their design, for example OceanStore[21, 20], Past[22], pStore[1], Pastiche[6], a system designed by Lillibrige et al.[15], Total Recall[3], PeerStore [13], Pastis[4] and Glacier[11].

Some of these systems are presented as peer-to-peer file-systems which makes them a little different from a backup system. In particular, persistence of data in backup systems is more important than read/write performances, while it is the opposite for file-systems. Most of them are still in prototyping phase, without real deployments. The main difficulty of this phase is that it is hard to have a large enough testbed, i.e. enough users trusting the system to use it on a daily basis. Our work with large size simulations could help developers to tune the parameters of these systems, like the repair threshold which is very difficult to set otherwise.

Another interesting optimization of the maintenance of a peer-to-peer backup system is presented in [10]. To decrease the probability of losing archives, their system measures the churn, i.e. the rate of departure of partners, and pro-actively creates new blocks at the same rate. Consequently, peer don't need to know exactly which partners have departed, relaxing the stress on the monitoring system.

6. CONCLUSION

This work has two main contributions. The first one is to prove the viability of a peer-to-peer backup system. We showed that, with a simple scheme based on lifetime estimation, a peer-to-peer backup system using erasure codes could achieve good performances in term of maintenance cost. Indeed, the number of repairs for normal users is far below the limit of feasibility imposed by the bandwidth of actual Internet home connections with almost one repair per archive in 200 days.

The second contribution is to provide a new criteria for the selection of partners. In our scheme, an older peer that has used the system for a long time, is considered more stable and more reliable than a newcomer. This assumption has often been verified in other systems. With this selection criteria, we achieve excellent performance for older peers, which are also the best contributors of the system. As a result, this may also be considered as a kind of tit-for-tat protocol. Reliable peers will be rewarded by excellent performances.

As future works, we plan to improve our simulations by allowing parameters to adapt more dynamically. For instance, the repair threshold might be changed depending on the peer context, its difficulties to find partners, the data that it needs to download, etc. We also plan to investigate more on the impact of temporary disconnections, in particular by delaying the repair to allow peers to come back in the system.

7. REFERENCES

[1] Batten, C., Barr, K., Saraf, A., and Trepetin, S. pStore: A secure peer-to-peer backup system. Tech. Rep. LCS-632, MIT Laboratory for Computer Science, 2001.

[2] Bhagwan, R., Savage, S., and Voelker, G. Understanding availability. In *IPTPS, Int'l Workshop On Peer-To-Peer Systems* (2003).

[3] Bhagwan, R., Tati, K., Cheng, Y., Savage, S., and Voelker, G. Total recall: System support for automated availability management. In *NSDI, Symposium on Networked Systems Design and Implementation* (2004).

[4] Busca, J.-M., Picconi, F., and Sens, P. Pastis: A highly-scalable multi-user peer-to-peer file system. In *Euro-Par'2005, International Euro-Par Conference* (2005).

[5] Bustamante, F., and Qiao, Y. Friendships that last: Peer lifespan and its role in P2P protocols. In *Int'l Workshop on Web Content Caching and Distribution* (2003).

[6] Cox, L., and Noble, B. Pastiche: Making backup cheap and easy. In *OSDI, Symposium on Operating Systems Design and Implementation* (2002).

[7] Cox, L., and Noble, B. Samsara: Honor among thieves in peer-to-peer storage. In *SOSP, Symp. on Operating Systems Principles* (2003).

[8] Dimakis, A. G., Godfrey, B., Wainwright, M. J., and Ramchandran, K. Network coding for distributed storage systems. In *INFOCOM, International Conference on Computer Communications* (2007).

[9] Duminuco, A., and Biersack, E. W. Hierarchical codes: how to make erasure codes attractive for peer-to-peer storage systems. In *P2P, International Conference on Peer-to-Peer Computing* (2008).

[10] Duminuco, A., Biersack, E. W., and En Najjary, T. Proactive replication in distributed storage systems using machine availability estimation. In *CoNEXT'07, 3rd International Conference on emerging Networking EXperiments and Technologies, December 10-13, 2007, New York, USA* (Dec 2007).

[11] Haeberlen, A., Mislove, A., and Druschel, P. Glacier: Highly durable, decentralized storage despite massive correlated failures. In *NSDI, Symposium on Networked Systems Design and Implementation* (2005).

[12] Jelasity, M., Montresor, A., Jesi, G. P., and Voulgaris, S. Peersim simulator, a peer-to-peer simulator.

[13] Landers, M., Zhang, H., and Tan, K.-L. Peerstore: Better performance by relaxing in peer-to-peer backup. In *P2P, Conference on Peer-to-Peer Computing* (2004).

[14] Le Fessant, F., Sengul, C., and Kermarrec, A.-M. Pacemaker: Tracking peer availability in large networks. Tech. rep., INRIA, 2008. RR-6594.

[15] Lillibridge, M., Elnikety, S., Birrell, A., Burrows, M., and Isard, M. A cooperative internet backup scheme. In *USENIX Annual Technical Conference, General Track* (2003).

[16] Maymounkov, P., and Mazieres, D. Kademlia: A peer-to-peer information system based on the XOR metric. In *IPTPS, Int'l Workshop On Peer-To-Peer Systems* (2002).

[17] Morales, R., and Gupta, I. AVMON: Optimal and scalable discovery of consistent availability monitoring overlays for distributed systems. In *ICDCS: Int'l Conf. on Distributed Computing Systems* (2007).

[18] Oualha, N., Önen, M., and Roudier, Y. Verifying self-organized storage with bilinear pairings. Tech. Rep. EURECOM+2311, Institut Eurecom, France, Jun 2007.

[19] Reed, I. S., and Solomon, G. Polynomial Codes Over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics 8*, 2 (1960), 300–304.

[20] Rhea, S., Eaton, P., Geels, D., Weatherspoon, H., Zhao, B., and Kubiatowicz, J. Pond: The oceanstore prototype. In *FAST, Conference on File and Storage Technologies* (2003).

[21] Rhea, S. C., and Kubiatowicz, J. Probabilistic location and routing. In *INFOCOM, International Conference on Computer Communications* (2002).

[22] Rowstron, A., and Druschel, P. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *SOSP, Symp. on Operating Systems Principles* (2001).

[23] Tian, J., and Dai, Y. Understanding the dynamic of peer-to-peer systems. In *IPTPS, Int'l Workshop On Peer-To-Peer Systems* (2007).