Data Durability in Peer to Peer Storage Systems*

Gil Utard
LaRIA
Université de Picardie Jules Verne
80000 Amiens,France
Gil.Utard@laria.u-picardie.fr

Antoine Vernois
Graal INRIA Project
LIP- École Normale Supérieure de Lyon
69364 Lyon Cedex 07, France
avernois@ens-lyon.fr

Abstract

In this paper we present a quantitative study of data survival in peer to peer storage systems. We first recall two main redundancy mechanisms: replication and erasure codes, which are used by most peer to peer storage systems like OceanStore, PAST or CFS, to guarantee data durability. Second we characterize peer to peer systems according to a volatility factor (a peer is free to leave the system at anytime) and to an availability factor (a peer is not permanently connected to the system). Third we model the behavior of a system as a Markov Chain and analyse the average life time of data (MTTF) according to the volatility and availability factors. We also present the cost of the repair process based on these redundancy schemes to recover failed peers. The conclusion of this study is that when there is no high availability of peers, simple replication scheme may be more efficient than sophisticated erasure codes.

1. Introduction

Today, Peer to Peer systems (P2P) are widely used mechanisms to share resources on Internet. Very popular systems was designed to share CPU (`Seti@home`, XtremWeb, Entropia) or to publish files (Napster, Gnutella, Kazaa, ...). In the same time, some systems was designed to share disk space (OceanStore, Intermemory, PAST, Farsite). The primary goal of such systems is to provide a transparent distributed storage service. These systems share common issues with CPU or files sharing systems: resource discovery, localisation mechanisms, dynamic point to point network infrastructure... But for sharing disk systems data lifetime is the primary concern. P2P CPU or file publishing systems can deal with node failures: the computation can be restarted anywhere or the published files resubmitted to the system. For disk sharing systems, node failure is a critical event: the stored data are definitively lost. So data redundancy and data recovery mechanisms are crucial for such systems.

In this paper we present a quantitative study of the usual data redundancy and repair schemes found in existing systems: replication and erasure resilient codes. After a presentation of some P2P systems and redundancy mechanisms, we introduce a characterisation of P2P storage system according to the volatility and availability of peers. We propose a stochastic model of such systems using Markov Theory which allows the study the data lifetime. We also consider the cost to recover lost data.

2. Peer to Peer system

Among peer to peer data systems, we distinguish two categories: P2P systems devoted to document publishing, and P2P systems devoted to storage and which integrate data survival mechanisms.

P2P systems are mainly characterised by an "over-network" or virtual backbone based on point to point connections leading to a loosely coupled graph. The aim of the over-network is mainly to furnish routing and localisation mechanisms.

Since the precursor Napster, lot of P2P file publishing systems have appeared. Gnutella [11] is one of them and is defined as a protocol specification to share documents. File localisation is done by a breath-search in the connection graph. Freenet [4, 3] is also a file publishing project where one of the primary goals is to insure anonymity of users (data producer or data consumer). It integrates cryptography of documents, auto adaptable routing, and a primitive replication mechanism to insure data survival of popular files.

Concerning peer to peer storage systems, one precursor is InterMemory [2, 6]. It is characterised by a complex redundancy scheme for data survival. PAST [5] is a joint project of the Rice University and Microsoft. In

*This Project (`http://www.ustorage.net`) is supported by the CNRS-INRIA-MENRT ACI GRID CGP2P grant.

PAST, nodes and files have a unique identifier in a common name space. Files are stored on node which have the closest identifier to the file identifier. Tolerant routing mechanism is PASTRY. Data survival is done by file replications. OceanStore [8, 1, 12] is a large project where data are stored in a set of collaborative server (long time survival, high speed connection) which are untrusted.

In this paper, we focus on this second class of peer to peer systems where data are disseminated on peers, and we study the efficiency of mechanisms used to insure data durability.

3. Redundancy mechanism

In distributed storage systems, redundancy is the base mechanism to protect data from node vanishing. The first redundancy scheme is replication: several copies of data are disseminated on different nodes. Replication is the mechanism used by PAST or Freenet. Let r be the replication factor: for $r = 1$ there is only one replica and the system tolerates only one node failure per data, if $r = 10$, the system tolerates 10 node failures per data.

A more sophisticated redundancy scheme is erasure code. In such a system a block of data of size B is fragmented into s smaller blocks of size $\frac{B}{s}$. In addition, r fragments of same size are generated and contain redundancy information about the initial data. These fragments are disseminated over the peer to peer network. Redundancy is such that any fragment can be recover from any combination of s fragments from the $s+r$ fragments. So this coding allows a tolerance factor equal to r. Replication is a particular case of erasure code where s is equal to 1. Erasure codes are used by the OceanStore project to maintain data-survival.

Let (s, r) be a redundancy scheme where block of data are divided into s fragments and where there are r fragments of redundancy. Ratio $\frac{s}{s+r}$ represents the useful space, i.e. ratio between size of the initial data and space used to store the data. For a fixed useful space, erasure code with a fragmentation factor s greater than 1 is *a priori* more efficient than simple replication because it allows a greater tolerance factor r. For instance, compare the $(1, 3)$ redundancy scheme with the $(4, 12)$.

An usual method to build erasure codes is the Reed Solomon encoding scheme [10]. Let B the initial data block to be encoded, B is considered as a vector $B = (b_j)_{j\in[1..s]}$ of dimension s where each b_j is a fragment of the initial data. The erasure code is the vector $E = (e_i)_{i\in[1..(s+r)]}$ of dimension $s+r$. The vector E is derived from B by the way of a matrix A of dimension $s \times (s+r)$: $E = AB$. Let A be decomposed by rows: $A = (A_i)_{i\in[1..(s+r)]}$. The matrix A is such that for each subset of s rows $(A_{i_j})_{j\in[1..s]}$, the rows are linearly independents. Thus, for any subset of s fragments $E' = (e_{i_j})_{j\in[1..s]}$ the initial data $B = (b_j)_{j\in[1..s]}$ is reconstructed in this following way: let $A' = (A_{i_j})_{j\in[1..s]}$ be the matrix build with the rows corresponding to the subset of fragments of E, the initial vector B is then equal to $A'^{-1}E'$.

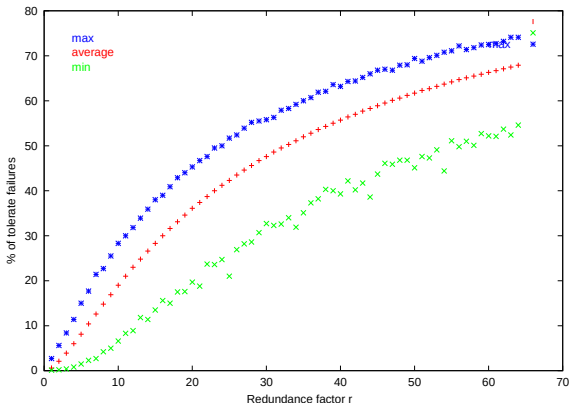


Figure 1. Percentage of peer failures before the loose of the first block for erasure code according to the redundancy factor r. The number s of initial fragments of the data block is equal to 16. There are 250000 blocks stored on 10000 peers.

Figure 1 shows the percentage of peers which can fail without loose of data according to the redundancy factor r. This result was obtained by the simulation of the distribution of 250000 blocks on 10000 peers where each block is divided in $s = 16$ fragments. For instance, with $r = 30$, we are able to tolerate up to 50% peer failures.

Whereas a big redundancy factor allows us to face to a big number of peer failures, this static mechanism does not provide good data survival. Consider for instance that peer lifetime follows an exponential deviate of parameter λ, more than half nodes probably fail after the average life time of one node.

To guarantee data survival, it is necessary to introduce a self repair mechanism which detects and rebuilds lost data. A repair mechanism is divided into two parts: The first part is the detection of node failure and the second is the reconstruction of the nodes' data. The reconstruction uses the redundancy mechanism previously described: dispersed redundancy fragments are collected to recover lost fragments on other peer(s).

4. A stochastic model for P2P storage systems

In this section, we present a stochastic model of the behaviour of peer to peer storage systems. The system is com-

posed of independent nodes which share their disk space where the fragments are stored. Nodes are free to leave the system at anytime. We assume that population in the system is stable: the rate of nodes leaving the system is equal to the rate of nodes entering the system.

4.1. Stochastic model for a peer

We assume also a homogeneous behaviour of peers. This behaviour is described by the stochastic automate depicted on Figure 2. In this automate, there are three states for

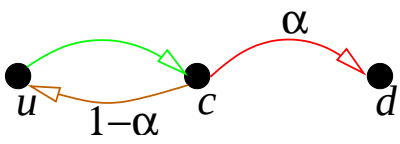


Figure 2. Node behaviour.

a peer node: connected (state c), temporarily unavailable or disconnected (state u), permanently unavailable or dead (state d). A node is not permanently connected to the system and may be disconnected for some time. The duration of a connection (resp. a disconnection) is given by a continuous random variable which follows an exponential deviate with an average lifetime equal to λ. (resp. μ). The average duration of a cycle connected/disconnected is given by $\lambda + \mu$. For the sake of presentation we set the unit of time equal to the cycle duration, i.e. $\lambda + \mu = 1$. The parameter λ is the *availability* factor of a node during its lifetime in the system. After a connection, a node can leave the system with a probability equal to α or stay in the system with a probability equal to $1 - \alpha$. The parameter α is the *volatility* factor of a node in the system.

The average lifetime denoted by τ of a node in the system, i.e. when it is in state c or u, is the average time of the first connection plus the product of average number of cycles connection/disconnection by the average duration of a cycle. So

$$\tau = \lambda + \sum_{k=0}^{k=\infty} (1-\alpha)^k \alpha(\lambda+\mu) = \lambda + \frac{1-\alpha}{\alpha}$$

(we fixed $\lambda + \mu = 1$). Since duration of states c and u follows exponential deviates and number of cycles follows a geometrical distribution, the lifetime of a node in the system is memoryless and follows a exponential deviate with an average duration equal to τ.

The parameters λ (availability) and α (volatility) allow us to characterize peer to peer systems (see Figure 3). For instance when λ is close to 1 and α is close to 0 we can consider the system to be a peer to peer server network like

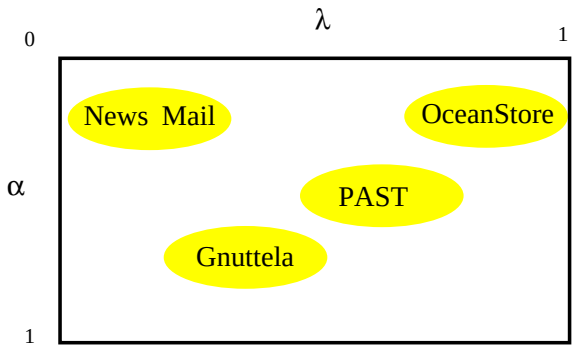


Figure 3. Characterisation of P2P systems according to their availability and their volatility.

OceanStore. When λ is smaller and α is close to 0 we can consider to be in presence of a system like a message service, i.e. peers are faithful but make short connection to the system. The worst case is when α is close to 1 and λ is small.

4.2. Block automate

For a given block of data, we describe its availability by the automate shows on Figure 4. The block is coded by $s+r$ fragments which are stored on $s+r$ distinct peers. We consider only these peers. A block is available when more than s peers are connected. A block of data is dead when more than r peers failed. A state is the couple (number of connected peer nodes, number of failed peer nodes). Arrows

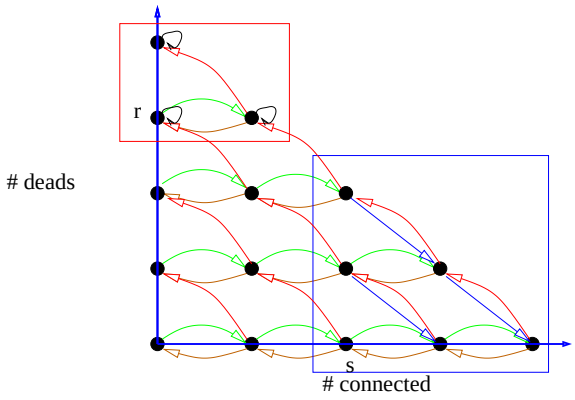


Figure 4. State transition for the availability of a block of data with a redundancy scheme $(s, r) = (2, 2)$.

correspond to the different transitions between states: connection of a peer (right arrows), disconnection of a peer

(left arrows) and failure of a peer (up arrows). States in the right box correspond to states where the data block is available (there are enough connected peers to rebuild the initial data), states in the upper box correspond to states where the data block is lost (more than r fragments are definitively lost and thus the initial data cannot be rebuild).

The repair process is represented by down arrows: the repair is done by a new peer which grabs s fragments of the block from connected peers and rebuild the lost fragment using the erasure code. So a repair can be done only when at least s fragments, i.e. peers, are connected. In our model, we assume that if a node fail, then there is enough space in the system to store the lost data, i.e. there is a peer not yet involved in the storage of the considered block which stores the recovered fragment.

4.3. Stochastic behavior of a block

If we considerer probabilities of state transitions, this automate describes a non recurrent Markov Chain with one absorbing state [7]. Let describe more formally the automate previously introduced. Let a_i^j be the state where i nodes are dead ($0 \leq i \leq s+r$) and j nodes are connected ($0 \leq j \leq s+r-i$). Probability for each state transition is determined in the following way.

Connecting: One peer is reconnecting the system, it is the transition from state a_i^j to state a_i^{j+1} ($0 \leq j < s+r-i$). Because all peers are independents the transition probability follows an exponential deviate with an average lifetime equals to $\frac{\lambda}{(s+r-j)}$.

Disconnecting: One peer is disconnecting the system, it is the transition from state a_i^j to state a_i^{j-1} ($0 < j \leq s+r-i$). Similarly to the reconnection, the transition probability follows an exponential deviate with an average lifetime equals to $\frac{\mu}{j}$.

Failure: One peer is leaving the system, it is the transition from state a_i^j to state a_{i+1}^{j-1} ($0 \leq i < s+r$, $0 < j \leq s+r-i$). The transition probability follows an exponential deviate with an average lifetime equals to $\frac{\tau}{(s+r-i)}$.

4.4. Dynamic repair transition

We employ the Markov Embedded Chain technique in order to estimate the transition probability of a repair transition. We consider that failed peers are repair in the order of their failure. The repair transition probability from state a_i^j to state a_{i-1}^{j+1} ($0 \leq i \leq s$,$s \leq j \leq s+r-i$) is the probability that when the oldest fail node is repaired, there are less than i failed nodes in the system.

The main problem is to determinate time to repair a node. We fix the *effective* time to rebuild a node (a.k.o. "cpu time") equal to t_e. However, the repair process is only possible when at least s nodes remain connected during the reparation process. We consider that when less than s nodes are connected, then the repair process is suspended. It restarts when s nodes are connected. So we have to evaluate what is the *real* time t_r to repair a node (a.k.o. "elapsed time").

We approximate t_r with the average fraction of time the repair process can be run (i.e. there are at least s nodes connected). Let f_r be this fraction, f_r is estimated in the following way: let d_i be the average lifetime to be in states a_i^j for all j such that $0 \leq j \leq s+r-i$ (there are i failed nodes) before one more node fails: $d_i = \frac{\tau}{s+r-i}$. Let c_i be the probability than at least s nodes are connected when there are i nodes failed. Connecting/reconnecting is a pure birth/death process, so c_i can be obtained by a simple product formulation. The fraction f_r is then estimated by:

$$f_r = \frac{\sum_i c_i d_i}{\sum_i d_i}$$

The real time t_r of the repair process is $\frac{t_e}{f_r}$.

Let $X(t)$ be the random variable representing the number of failed node remaining when the oldest failed node is repair at time t (pure death process). We estimate the probability of a repair transition from state a_i^j to state a_{i-1}^{j+1} ($s \leq j \leq s+r-i$, $i < s$) by $P(X(t_r) < i)$.

5. MTTF of data blocks

In the following we present a quantitative study of the modeled P2P storage system with automatic repair process. This quantitative study allow us to estimate what is the MTTF (*mean time to failure*) of data block in such a system, i.e. the average lifetime of a data block in the system.

Thanks to our Markov model, the MTTF can be estimated classically by a discretisation of time and considering the resulting transition probability matrix P where the absorbing state is removed. Then we compute the vector $E = (I-P)^{-1} \times U$ where I is the identity matrix and U the unit vector. The elements E_i gives the average lifetime starting from state i.

Figure 5 shows the estimated MTTF according to the availability rate of peers (λ), thus for three redundancy schemes with the same useful space ((7,21), (4,12) and (1,3)[1]) and for two volatility parameters: $\alpha = 0.1$ and $\alpha = 0.3$ (for high volatility factors, the MTTF is too low). The time fixed for the repair process is $t_e = 1$, i.e. the time for one period of connection/disconnection (we observed a

[1] The redundancy scheme (1,3) corresponds to the replication scheme.

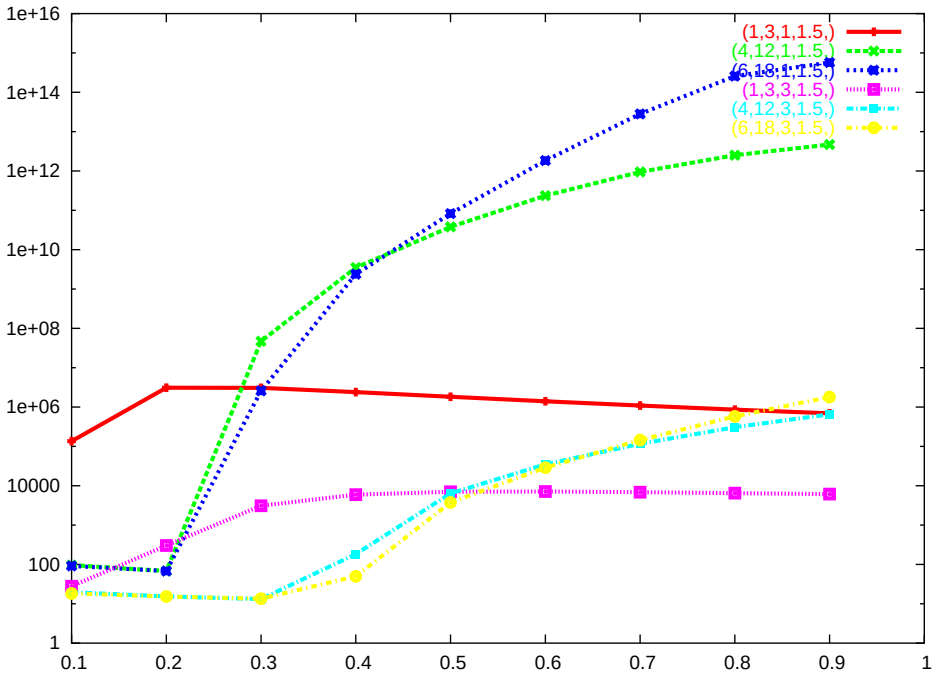


Figure 5. MTTF according to the availability of peers (λ).

similar behaviour for other repair process time). For a system with high availability (λ close to 1), MTTF is better with s big as expected. But for a system with low availability, we observe a better MTTF for the replication scheme.

This suggests that for a peer to peer system with lower availability of peers, the replication scheme is better than erasure code. Figure 6 shows the MTTF according to the fragmentation parameter s for a fixed useful space ratio (25%), and for different volatility and availability parameters.

This phenomena may be explain by considering the frequency of each state. Figure 7 shows the relative frequency of each state for a (4,12) erasure code which is obtain using Markov Theory. Darker state are the more frequent state during the life of a block. These frequency was plot for $\alpha = 0.3$ and different availability parameters (0.4, 0.5 and 0.7). The vertical line is the limit number of connected peers which are necessary to rebuild a failed peer. When the availability is reduced, we see that more frequent states are close to this limit. This means that the state of the block is more often in a state where the repair process is suspended, so the real time to repair is greater, probability of not recovering the block of data (which determines MTTF) increases.

6. Data recover costs

In the previous part, we estimated the MTTF of a peer to peer system characterized by (α,λ) based on a redundancy scheme (s, r) which integrates a failure repair mechanism. In this part, we will discuss about costs generated by this repair mechanism.

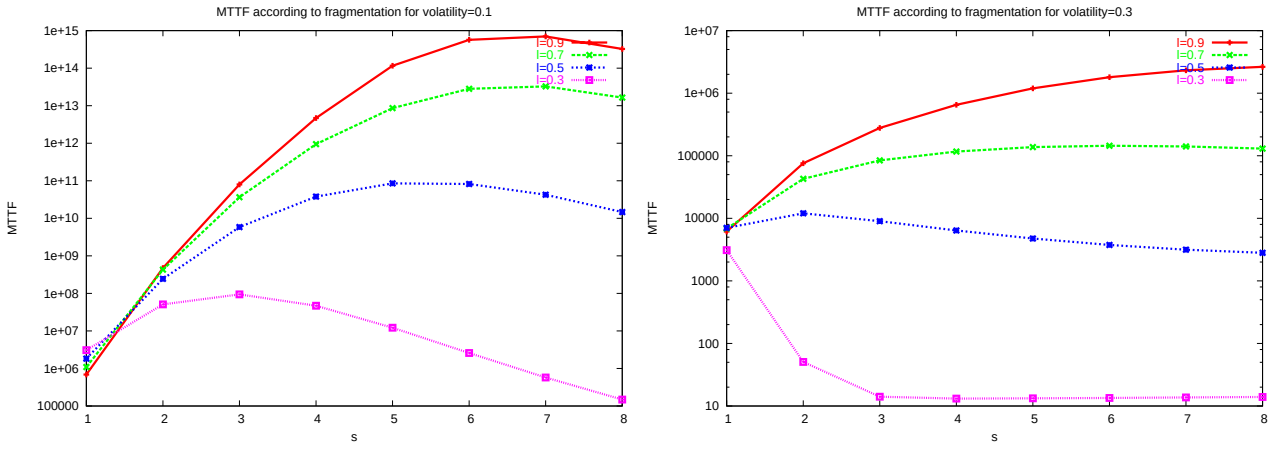
One of our hypothesis is that the repair mechanism is able to rebuild data of a failed node in a fixed time t_e. To rebuild one fragment in a (s, r) redundancy erasure code, s fragments must be retrieved from alive nodes. Lets say that each node hosts 10000 fragments of 4KB (i.e. 40MB per peer), and let s be equal to 8, then $10000 * 4 * 8 = 320$MB have to transit on the network to repair one failed peer. The peer to peer system must deal with simultaneous failed peers, so the repair process is limited by the capacity of the peer to peer network.

Let $T_x = \frac{1}{\tau}$ be the average rate of node failure where τ is the average lifetime of a node. In our model

$$T_x = \frac{1}{\lambda + \frac{1-\alpha}{\alpha}}$$

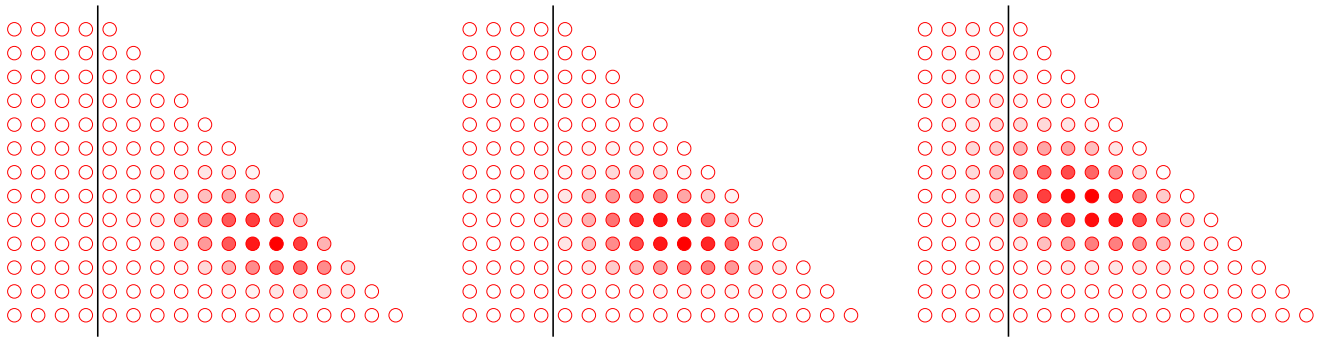
T_x increases with α, and then the number of simultaneous failed peers to repair increases in the same order.

Figure 8 is the average peer lifetime according to the volatility α. We can see these plots as the average number of time period ($\lambda + \mu$) a node stays in the system before leaving it. This means that $\frac{1}{Tx}$ peer must be repair per cycle on average. The amount of data to communicate in the network is proportional to $\frac{1}{Tx}$. Using an erasure code with a fragmentation factor equals to s implies that the amount of data communicated in the network is multiplied by s in a



(a) $\alpha = 0.1$ (b) $\alpha = 0.3$

Figure 6. MTTF according to the fragmentation factor s for different λ (l) and α.



(a) $\lambda = 0.7$ (b) $\lambda = 0.5$ (c) $\lambda = 0.4$

Figure 7. State frequency according of a $(4, 12)$ redundancy scheme for different availability factors λ ($\alpha = 0.3$).

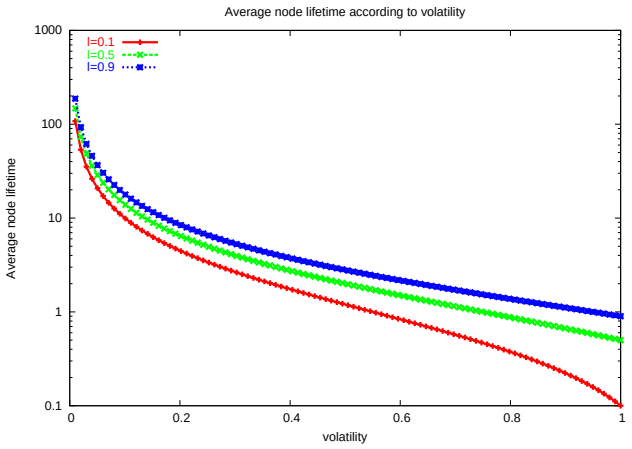


Figure 8. Average lifetime of node according to α for different λ.

cycle. For instance is $Tx = 8$ and $s = 8$, then the amount of data communicated in the network in a cycle is equal to the volume of data stored!

6.1. Conclusion

In this paper, we presented a quantitative study of data survival in Peer to Peer Storage Systems. We characterized Peer to Peer systems according to a volatility factor (peer are free to leave the system at anytime) and to an availability factor (peer are not permanently connected to the system).

We compared the two main redundancy schemes, replication where data are duplicated on different peers, and erasure codes where data are first divided into s fragments and where r fragments of redundancy are added to tolerate failure. We estimated the data lifetime of these redundancy scheme by a Markov Chain model. Whereas the erasure coding scheme is *a priori* more efficient than simple replication, a result of this study is that simple replication is better when there is no good availability of peers: erasure codes require high availability of peers to be efficient.

For repair cost point of view, we also shown that erasure codes increase the amount of communication to recover failed peer: s times the amount of data stored in the failed peer must be accessed to rebuild the lost data. A large part of the network capacity must be used to maintain data integrity.

In fact, contrarily to conventional wisdom, erasure codes like Reed Solomon are not the *cure-all* to insure long data lifetime in peer to peer storage systems: it is efficient only for peer to peer system with highly available peers like OceanStore, where peers are servers hosted by some Internet Providers. Our future work is to design and analyse other redundancy scheme mechanisms which may be better than simple replication for data survival and erasure code for the cost of the repair mechanism. We can imagine new redundancy scheme which mixes replication and erasure code to combine the low cost of replication and to increase the efficiency of erasure code because of a better availability of fragments. We plan also to study the Tornado [9] method which employs a sparse hierarchical redundancy scheme which may reduce the needs for highly available peers.

References

[1] David Bindel, Yan Chen, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Christopher Wells, Ben Zhao, and John Kubiatowicz. Oceanstore: An extremely wide-area storage system. Technical Report Technical Report UCB CSD-00-1102, U.C. Berkeley, May 1999.

[2] Yuan Chen, Jan Edler, Andrew Goldberg, Allan Gottlieb, Sumeet Sobti, and Peter Yianilos. A prototype implementation of archival intermemory. In *Proceedings of the Fourth ACM International Conference on Digital Libraries*, 1999.

[3] Ian Clarke. A distributed decentralised information storage and retrieval system, 1999.

[4] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, 2000.

[5] P. Druschel and A. Rowstron. PAST: A large-scale, persistent peer-to-peer storage utility. In *Procedings of HOTOS*, pages 75–80, 2001.

[6] A. Goldberg and Peter N. Yianilos. Towards an archival intermemory. In *Proceedings of IEEE Advances in Digital Libraries, ADL 98*, pages 147–156, Santa Barbara, CA, 1998. IEEE Computer Society.

[7] Leonard Kleinrock. *Queueing system*, volume Volume I: theory. Wiley-Interscience Publication, 1975.

[8] John Kubiatowicz, Divid Bindel, Yan Chen, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westly Weimer, Christopher Wells, and Ben Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of ACM ASPLOS*. ACM, November 2000.

[9] M. Luby, M. Mitzenmacher, A. Shokrollay, and D. Spielman. Efficient Erasure Correction Codes. *IEEE Trans. on Information Theory*, 47(2), February 2001.

[10] James S. Plank. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems. *Software Practice and Experience*, 27(9):995–1012, 1997.

[11] Matei Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *Proceedings of International Conference on Peer-to-peer Computing*, August 2001.

[12] Chris Wells. The oceanstore archive: Goals, structures, and self-repair. Master's thesis, University of California, Berkeley, May 2001.