Peer-to-Peer Storage Systems: a Practical Guideline to be Lazy

Frédéric Giroire, Julian Monteiro, Stéphane Pérennes

Peer-to-Peer Storage Systems: a Practical Guideline to be Lazy

Frédéric Giroire and Julian Monteiro and Stéphane Pérennes
MASCOTTE joint project INRIA / I3S (CNRS, Univ. of Nice-Sophia), France
Email: {firstname.lastname}@sophia.inria.fr

***Abstract*—Distributed and peer-to-peer storage systems are foreseen as an alternative to the traditional data centers and in-house backup solutions. In the past few years many peer-to-peer storage systems have been proposed. Most of them rely on the use of erasure codes to introduce redundancy to the data. This kind of system depends on many parameters that need to be well tuned, such as the factor of redundancy, the frequency of data repair and the size of a data block. In this paper we give closed-form mathematical expressions that estimate the system average behavior. These expressions are derived from a Markov chain. Our contribution is a guideline to system designers and administrators to choose the best set of parameters. That is, how to tune the system parameters to obtain a desired level of reliability under a given constraint of bandwidth consumption. We confirm that a lazy repair strategy can be employed to amortize the repairing cost. Moreover, we propose a formula to calculate the optimal threshold value that minimizes the bandwidth consumption. Finally, we additionally discuss the impact of different system characteristics on the performance metrics, such as the number of peers, the amount of stored data, and the disk failure rate. To the best of our knowledge this is the first work to give close-form formulas to estimate the bandwidth consumption for a lazy repair, and the loss rate taking into account the repair time.**

I. Introduction

Peer-to-peer systems are an interesting alternative to obtain a storage solution with high reliability at low cost. Some backup solutions, e.g., data centers and high-end NAS appliances are highly reliable, but also tend to be very expensive. The advantages of using a peer-to-peer network for data archival is that, by nature, it can distribute the information into different locations and has a high potential to be scalable.

Many systems have been proposed, e.g., CFS, Farsite, OceanStore, PAST, Glacier, TotalRecall, or Carbonite (see [5] and references within). Also, there is an effort by the open source community to create a global storage cloud, the Tahoe-LAFS [16] project. Commercial companies, like Wuala [17] and Ubistorage [14] , exploit that technology to deliver a reliable backup solution.

The key concept of peer-to-peer storage systems is to introduce redundancy to data and distribute it among peers in the network. The addition of redundant data could be done by trivial *Replication* [13], [3], in which copies of data are sent to different nodes in the system; or be based on *Erasure Codes* [10], [15], such as Reed Solomon and Tornado, as used by some RAID schemes.

When using Erasure Codes, the original user data (e.g. files, raw data, etc.) is cut into *data-blocks* that are in turn divided into s initial *fragments* (or pieces) of equal size. The encoding scheme produces $s + r$ fragments that can tolerate r failures (see Figure 1). In other words, the original data-block can be recovered from any s of the $s + r$ encoded fragments. In a peer-to-peer storage system, these fragments are then placed on $s + r$ different peers of the network.

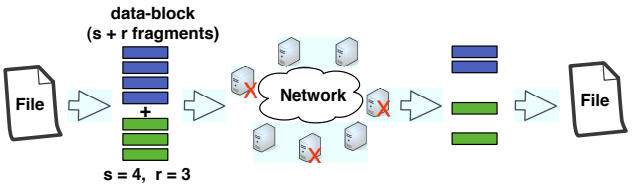

Fig. 1. Files or raw data are cut into data-blocks. Each data-block is divided into s initial fragments, to which r fragments of redundancy are added. Any s fragments among s+r are sufficient to recover the original data-block.

To keep a durable long-term storage, the system must be capable to maintain, despite disk failures, a minimum number of fragments of each block present in the network. We study the case of *reactive maintenance* and *lazy repair*, that is, when the number of redundant fragments drops below a certain level, namely r_0, the block is reconstructed.

A fundamental question for such systems is how to choose the basic set of parameters, such as s, r, and r_0, to obtain an efficient utilisation of bandwidth for a desired level of reliability?

In this paper, we analyse the steady-state of a storage system based on erasure codes. Our contribution is a practical guide to choose the best set of system parameters to obtain a desired level of reliability under a given constraint of bandwidth consumption, or vice-versa. To the best of our knowledge this is the first work to give close-form formulas to estimate the bandwidth consumption, and the loss rate taking into account the repair time. These formulas are derived from a simplified Markov chain model. They give a good intuition of the system dynamics and of the impact of its parameters. We considered two scenarios: in the first one we study the trade-off between the bandwidth consumption and the loss rate for a *fixed storage space*; in the second scenario, for a given reliability, we show how to *provision the space overhead* to obtain an optimal bandwidth usage.

The remainder of this paper is organized as follows: after presenting the related work, we describe the system characteristics in the next section. In Section III we present a simplified Markov Chain Model along with a compilation of closed formulas to estimate the system metrics in Section IV. In Section V, we explain how to choose the system parameters.

Related work. Many works study the use of erasure codes schemes to add reliability to data. In [15], Weatherspoon

and Kubiatowicz show that, in most of the cases, erasure codes use an order of magnitude less bandwidth and storage space than replication to provide similar system durability. Lin et al. in [9] and Rodrigues and Liskov in [12] also study the trade-offs of using erasure codes for different peer availabilities. In Total Recall [2] the authors propose the lazy repair mechanism and study the system behavior for different peer availabilities. Datta and Aberer in [7] study analytical models for different lazy maintenance strategies. Dimakis et al. in [8] compare different erasure codes against a lower bound, but without the lazy repair approach. In [6], it is shown that the disk failures induce important variations around the mean bandwidth consumption. However, there is no discussion on the choice of the system parameters. Our work differentiate from those as we analyse the *bandwidth efficiency* of erasure codes using *lazy repair*, and we explore the parameter's space to give a simple procedure to estimate their values.

II. System Description

The detailed characteristics of the studied P2P storage system are presented in this section.

Data Redundancy. Erasure Code schemes [10] are used to introduce data redundancy in the system. The user data is cut into data-blocks (or block, for short). Each data-block is divided into s initial pieces. Then r pieces of redundancy are added, in such a way that the initial block can be reconstructed from any subset of s pieces among the $s+r$. The pieces are then sent to $s+r$ different peers at random. The stretch factor (or space-overhead) is defined as $(s+r)/s$.

Peer Availability. It is assumed that the peers stay connected almost all the time into the system. To avoid the problem of transient failures and deal with short-time churn, a peer is considered lost only if it has left the system for a period longer than a given timeout [12].

Peer Failures. In our model a peer failure represents a disk crash or a peer that definitively leaves the system. In both cases, it is assumed that all the data on the peer's disk are lost. Following most works on P2P storage systems [1], [11], peers fail independently according to a memory-less process. For a given peer, the probability to fail at any given time step is $\alpha = 1/\text{MTTF}$. The probability for a peer to be alive after T time steps is $(1-\alpha)^T$.

Data Repair. The system needs to continuously monitor the block's redundancy level to decide if a repairing process (namely *reconstruction*) needs to be done. The reconstruction is done in three consecutive phases: first, *retrieval*, the peer in charge of the reconstruction has to download s fragments among the remaining block's fragments; secondly, *recoding*, that recreate the data-block; and the third, *sending*, in which the reconstructed missing fragments are sent to different peers.

Reconstruction Strategy. Different reconstruction strategies can be considered. Delaying the reconstruction, i.e., waiting for a block to lose more than one fragment before rebuilding it, amortizes the costs of bandwidth usage over several failures. Hence, we study a *threshold based reconstruction policy* in this paper. When the number of fragments of a block drops to a threshold value r_0, the reconstruction starts. Note that, when r_0 is set to $r-1$, the reconstruction starts as soon as a first piece is lost. This special case is called *eager policy* and the induced cost to reconstruct is very high, because it is necessary to transfer $s+1$ fragments to reconstruct only 1 fragment. Setting a low value for r_0 decreases the number of reconstructions (as the reconstruction starts only after that $r-r_0$ pieces are lost), but increases the probability to lose a block.

TABLE I
Summary of main notations.

N	# of peers
D	amount of data to store, in bytes
s	# of initial fragments of a block
r	# of redundancy fragments
r_0	reconstruction threshold value
l_f	size of a fragment, in bytes
l_b	initial size of a block, in bytes ($l_b = s \cdot l_f$)
B	total # of blocks in the system ($B = D/l_b$)
$\overline{F}$	total # of available fragments at steady state
$MTTF$	peer mean time to failure
α	prob. for a disk to failure during a time step ($\alpha = 1/MTTF$)
$\delta(i)$	probability for a block at level i to lose one fragment
Θ	time steps to reconstruct all blocks after a disk failure
θ	average time steps to reconstruct one block
γ	prob. for a block to be reconstructed at a time step ($\gamma = 1/\theta$)
τ	time step of the model

Size of the Studied System We study the system's characteristics in the steady-state. Moreover, the number of peers N, and the amount of data stored in the system D, is kept constant over the time. Crashed disks reappear empty.

III. Markov Chain Model (MCM)

As shown in [6], the system averages can be precisely modeled by a finite discrete time Markov Chain. The states (shown in Figure 2) represent the level of redundancy of one block in the system.

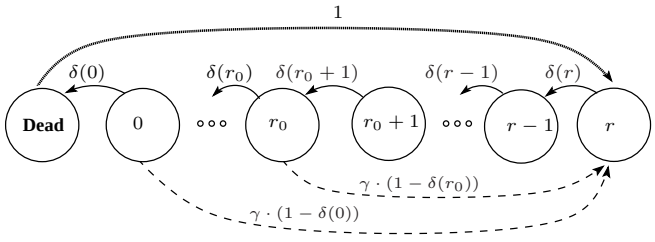

Fig. 2. Markov chain modeling the behavior of one block. Solid and dashed lines represent failure and reconstruction events,respectively. Loops are omitted. Dead blocks are re-injected in the system with probability 1.

The chain has $r+2$ states, that are the r levels of redundancy of a block b, plus a level 0 (no more redundancy), and a *dead* state. Three different kinds of states can be distinguished:

- **Non-Critical:** when $r_0 < r(b) \leq r$;
- **Critical:** when $0 \leq r(b) \leq r_0$;
- **Dead:** when the block has less than s fragments,

where $r(b)$ is the number of remaining redundancy fragments of a block b. A block can be affected by two kinds of events: peer failures and reconstructions. The probability for a block at level i to lose one fragment during a time step is denoted

by $\delta(i)$ and is approximated by $\delta(i) = (s+i)\alpha$, (recall that α is the probability for a peer to experience a failure during the time step). When a block becomes critical, i.e., the remaining redundancy $r(b) \leq r_0$, the reconstruction starts. The average duration of a reconstruction is noted θ. At each time step, a critical block has a probability $\gamma := 1/\theta$ to be rebuilt. In that case it goes to the top, state r.

In our model, due to the stability assumption (the amount of data is constant), a dead block is replaced immediately. This purely formal assumption does not affect the system behavior because dead blocks are rare events, but it makes the analysis more tractable.

IV. Estimating System Metrics

Modeling this system with a Markov Chain Model is very useful to describe the system average behavior. In this section we present some explicit expressions to estimate the system main metrics and the peek of bandwidth when there is a peer failure. These expressions give an intuition of the system behavior in function of its parameters. We provide approximations for ratios $\alpha/\gamma \ll 1$, which is the case for practical systems where the block reconstruction process is much faster than the peer failure rate.

Stationary distribution. The finite Markov chain presented above is irreducible and aperiodic. Hence, the probability to be in a state converges towards a unique stationary distribution denoted by P, where $P(i)$ is the stationary probability to be in state i. In a system where the blocks are distributed uniformly at random and the peers fails independently, we can say that each state in the chain represent the fraction of blocks at that state. In that case, it is very easy to derive simple closed-formulas that estimate the distribution of the blocks' redundancy level. The stationary distribution of that chain can be computed by the stability equations as follows:

$$P(i) = \begin{cases} \frac{\delta(i+1)\cdot P(i+1)}{\delta(i)}, & \text{if } r_0 < i \leq r \\ \frac{\delta(i+1)\cdot P(i+1)}{\delta(i)+\gamma\cdot(1-\delta(i))}, & \text{if } 0 \leq i \leq r_0 \\ \delta(0)P(0), & \text{if } i = dead \end{cases}$$

All probabilities can be expressed in function of $P(r)$, which in turn is computed by the normalization $\sum_{i=0}^{r} P(i) + P(dead) = 1$. Then the fraction of fragments at level $P(r)$ can be simplified as $P(r) \approx \frac{1}{(s+r)\cdot(H_{s+r}-H_{s+r_0})}$, where $H_n = \sum_{k=1}^{n} \frac{1}{k}$ is the harmonic number of n. Note that, hereafter we use the approximation $\ln(n) \approx H_n$. The fraction of blocks at each Non-Critical state i is then $P(i) \approx \frac{(s+r)\cdot P(r)}{(s+i)}$, which evaluates as

$$P(i) \approx \frac{1}{(s+i)\cdot \ln(\frac{s+r}{s+r_0})}, \text{ if } r_0 < i \leq r. \qquad (1)$$

Distribution of blocks' redundancy level. If every block is at maximum redundancy, the total number of fragments stored in the system is $F = B\cdot(s+r)$. However, at the steady-state this is not true. In that case, the average redundancy level, $\mathbb{E}[P]$, is in between r and r_0. Figure 3 illustrates the average number of blocks in each redundancy level. Note that the number of blocks in the Non-Critical states (levels 8 to 11) are not evenly distributed. We can then estimate the number of available fragments at the steady-state, $\overline{F} = B \cdot \mathbb{E}[P]$, which is approximated by $\overline{F} \approx B\cdot(s + \frac{r+r_0}{2})$.

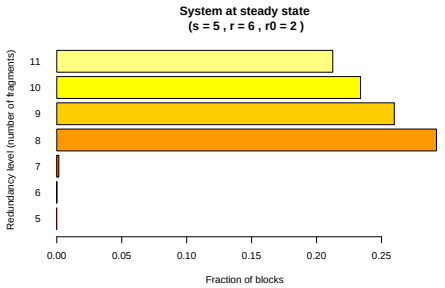

Fig. 3. Distribution of blocks' redundancy level at the steady state.

A. Estimating the Bandwidth Consumption

To estimate the bandwidth usage by reconstructions, we first need to define the repair bandwidth inefficiency $\epsilon(i)$, as the amount of data that need to be transferred to reconstruct i missing fragments: $\epsilon(i) = (s + i - 1)\cdot l_f$, where l_f is the size of a fragment. That is, the peer in charge of the reconstruction must download s fragments from other peers and then send $i-1$ reconstructed fragments (assuming that the peer in charge keeps a fragment). When a block needs to be reconstructed, the number of missing fragments is in most of the cases $r - r_0$. Sometimes it could be a little bit larger if two fragments were lost during the reconstruction, which is rare when the ratio $\alpha/\gamma \ll 1$.

Average bandwidth consumption. At the steady-state the number of blocks that finish the reconstruction at each time step is equal to the number of reconstructions that start (by a cut argument in the chain). Then the average bandwidth consumption comes straightforward from the transition of the state $r_0 + 1$ to the state r_0. That is, the fraction of blocks that goes from the last *Non-Critical* state towards r_0, given by $\delta(r_0 + 1)\cdot P(r_0 + 1)$.

For a system with B blocks, the average number of blocks finishing the reconstruction is $R_{avg} = B\cdot(s + r_0 + 1)\cdot\alpha\cdot P(r_0 + 1)$. For each block, the amount of information to be transferred is, in most of the cases, $\epsilon(r - r_0)$. If the reconstructions are uniformly distributed, the average bandwidth consumption per peer is $BW_{avg} \approx \frac{R_{avg}\cdot\epsilon(r-r_0)}{N\cdot\tau}$, which evaluates as

$$\boxed{BW_{avg} \approx \frac{B\cdot\alpha}{N\cdot \ln(\frac{s+r}{s+r_0})\tau}\cdot(s + r - r_0 - 1)\cdot l_f.} \qquad (2)$$

Note that BW_{avg} does not depends on the reconstruction rate γ. This expression is valid for systems with ratio $\alpha/\gamma \ll 1$ (see Figure 4 and corresponding discussion).

Peek of bandwidth consumption. It is know that the dynamics of a system with many blocks is not smooth at all. There

are peeks of resource consumption when a peer fails, because many fragments are lost at the same time and trigger many reconstructions (see [6]). The number of blocks that start reconstruction when a peer fails is $R_{start} = \frac{\overline{F}}{N} \cdot P(r_0 + 1)$, where $\overline{F}/N$ is the average number of fragments per disk, and $P(r_0+1)$ is the fraction of these fragments that belongs to blocks that are at the last Non-Critical state (i.e., need to start the reconstruction). The amount of data transfer induced by these reconstructions is $Q_{peek} = R_{start} \cdot \epsilon(r - r_0)$ and can be approximated as

$$Q_{peek} \approx \frac{B \cdot (s+r)}{N \cdot (s + r_0 + 1) \cdot \ln(\frac{s+r}{s+r_0})} \cdot (s + r - r_0 - 1) \cdot l_f.$$

Q_{peek} can be used to calculate the average time to reconstruct the data of a failed peer, which in turn can be used to re-estimate the reconstruction rate γ for a given bandwidth capacity.

B. Estimating the Data Loss Rate

The evaluation of $P(dead) = \delta(0) \cdot P(0)$ gives the fraction of blocks that are lost per time step τ. Hence in a system with B blocks the data loss rate can be calculated as $LossRate = B \cdot P(dead)/\tau$. When $\alpha/\gamma \ll 1$ it is closely approximated as

$$LossRate \approx \frac{B}{(s + r_0 + 1) \cdot \ln(\frac{s+r}{s+r_0})\tau} \cdot \frac{(s+r_0)!}{(s-1)!} \cdot \left(\frac{\alpha}{\gamma}\right)^{r_0+2}. \quad (3)$$

C. Discussion on the system behavior

In this section we discuss the performance metrics for different system characteristics. We first give an example of a medium size network composed by $N = 500$ peers and $D = 20$ TB of data to be stored. The MTTF of peers is set to 1 year. This value is less than a typical time-span of warranties applied by major hardware vendors, which is 3 years. Indeed, this value is a conservative choice and comprises the probability of other hardware failures and of software maintenance.

Lets choose $s = 16$, $r = 16$ and $r_0 = 8$ (we discuss about this choice in the next section) and a fragment size $l_f = 320$ KB. We obtain a block size $l_b = s \cdot l_f = 5$ MB and the total number of blocks $B = D/l_b = 2^{22}$. The initial amount of data per disk is 82 GB (at the steady state it is 72 GB). For such parameters, the average bandwidth usage per peer $BW_{avg} \approx 57.8$ kbps. When a peer fails, the total amount of data to be transferred $Q_{peek} \approx 246$ GB (504 MB per peer). For a provisioned reconstruction time $\theta = 12$ hours the $LossRate \approx 5.7 \cdot 10^{-8}$ per year.

System size: Systems with different sizes can be analyzed in two scenarios: the first is when N is larger and the amount of stored data in the system D remains constant, in that case BW_{avg} and Q_{peek} decreases inversely with N. However, $LossRate$ does not change. In other words, for the same amount of data, a larger system behaves more smoothly. The other scenario is when N is larger and D increases at the same rate, i.e., the amount of stored data per peer $d = \frac{D}{N}$ remains constant. Then, we remark that BW_{avg} and Q_{peek} also remain constant. But, in this case $LossRate$ increases when the system is larger (it depends on the absolute number of blocks in the system, $B = D/l_b$, we discuss the choice of this parameter in the next section).

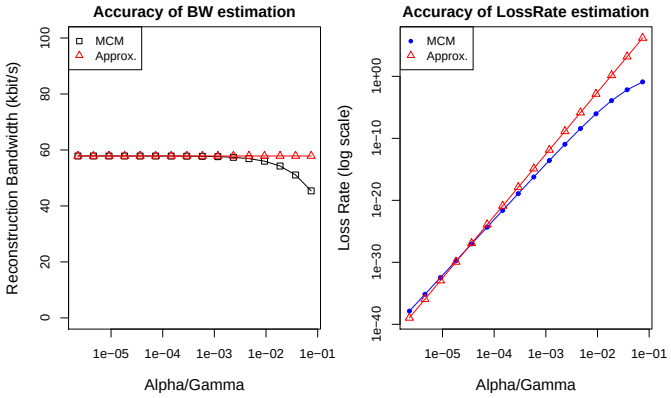

Fig. 4. Accuracy of estimations for different ratios α/γ.

Disk failure rate (α): The BW_{avg} is impacted linearly by α, that is, more failures means more bandwidth usage. The probability to lose data increases exponentially with α. As expected, Q_{peek} is not impacted by α. Nonetheless, when α is higher, the peeks of bandwidth occur more often.

Validations. Figure 4 shows the accuracy of Equations (2) and (3) compared to the MCM for different ratios of α/γ. Note that for values of $\alpha/\gamma < 10^{-3}$ the results obtained by the equations are very close to the MCM. For such values of α/γ our experimentation with different parameters confirmed the accuracy of the equations. Moreover, in [6], it is shown that the values obtained by the MCM closely match the results obtained by simulations. Hence, for space reasons these results are not presented here.

V. How to Set the System Parameters

The choice of the system parameters depends on multiple constraints, for instance, the storage space-overhead, the bandwidth consumption, the desired level of reliability, etc. In this section, we propose a methodology to choose the main system parameters B, l_f, s, r, r_0, for a desired reliability (probability to lose data) or a given limit on the bandwidth consumption.

In brief, we start defining a suitable value for s and l_f, which depends on the system architecture and usage. Then, if the space-overhead ($\frac{s+r}{s}$) is fixed, we choose the best r_0 for the given constraints. Otherwise, we first choose r_0 and then calculate the best value of r to minimize the bandwidth consumption.

A. Determining the block size (l_b)

The total number of blocks in the system is defined by $B = D/l_b$. For a given amount of data D, how do we choose

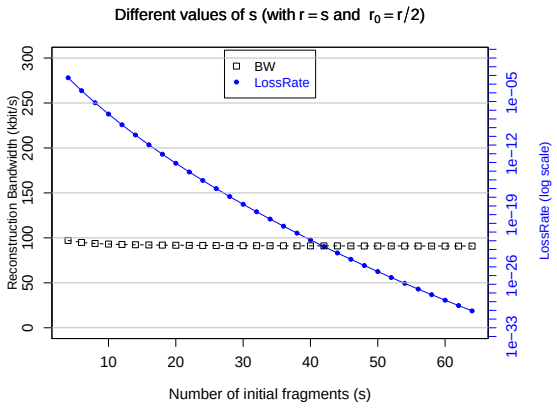

Fig. 5. System with fixed number of blocks B, increasing s and decreasing l_f. The redundancy r and reconstruction threshold r_0 are chosen as a factor of s, respectively, $r = s$ and $r_0 = s/2$.

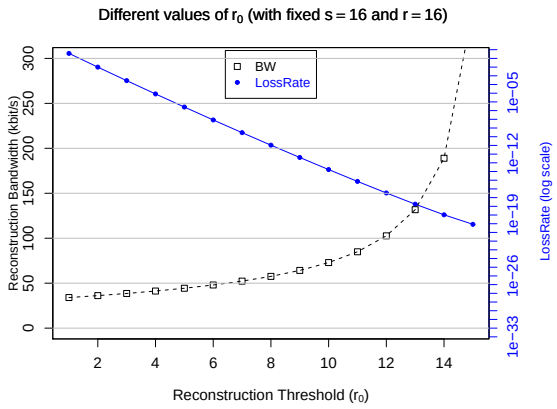

Fig. 6. System with fixed space-overhead of 2. The parameters $s = r = 16$. The choice of r_0 depends on the desired reliability (prob. of data loss) and the bandwidth consumption.

l_b? Similarly, knowing that $l_b = s{\cdot}l_f$, how do we choose s and l_f?

In this discussion we assume that r and r_0 are defined as a factor of s, that is, $r = k{\cdot}s$ and $r_0 = k_0{\cdot}s$. Hence, increasing s means increasing r and r_0 proportionally. By rewriting r and r_0 in the Equation (2) we note that BW_{avg} does not almost depend on the ratios between B, s and l_f, but mainly on the constant D. Hence, the choice of the block size is based only on the $LossRate$ equation.

Architectural issues: from a theoretical point of view, to obtain lower values of $LossRate$, B should be as small as possible, and therefore l_b as big as possible.

However, in practice we can deduce a lower bound for B based on R_{start}, the number of blocks that start the reconstruction when a peer fails. To balance the load among peers, every peer should process the reconstruction of at least one block. Hence, $R_{start} \geq N$, which evaluates to the constraint $B \geq N^2$.

The choice of l_b also depends on the main usage of the storage system. Two main groups of usage can be distinguished. For an *archival usage*, in which the access to the stored data is very rare, the block size l_b could be very large. Conversely, in a *filesystem usage*, e.g, Pastis [4], that supports continuous read and write operations, it is interesting to have a very small block size. So, the overhead of accessing and modifying a block is low.

The choice of s: For a fixed $l_b = s{\cdot}l_f$, s should be as large as possible and l_f as small as possible. Figure 5 shows the bandwidth consumption and probability to lose data (in log scale) when using the MCM, for a system with fixed B, increasing s from 4 to 64, and proportionally decreasing l_f. In this experiment, $k = 1$ and $k_o = 1/2$. As expected, the results show that larger values of s do not impact the bandwidth consumption, whereas the probability to lose data decreases exponentially, as stated in Equation (3).

But note that the size of l_f should not be too small. Some practical limits impose a value of at least 4 KB, which is the common value of file system's block size. Moreover, the amount of metadata, m_f, that should be kept is linearly dependent on the number of fragments of a block. In a practical system $m_f \ll l_f$, if not, the metadata takes up an important space that could be used to achieve more redundancy.

Another factor on the choice of s is the encoding and decoding rate of the erasure codes. Some implementations of the classic Reed-Solomon use words of length *one byte* to improve the efficiency (they work on the Galois Field $GF(2^8)$), which leads to the practical limitation $s+r \leq 256$. The overhead of the encoding is of order $O(s{\cdot}r)$. Very high values of s could impact negatively the encoding throughput. For instance, when s and r are large (e.g., $s = r = 128$) the encode throughput could be as low as 20 Mbps (on a Core 2 Duo 2.16Ghz).

B. Determining the reconstruction threshold (r_0)

For a given s, the choice of the threshold value r_0 depends on two factors: the desired reliability and the bandwidth capacity. The reliability can be calculated using Equation (3). It is sufficient to find the smallest r_0 that matches the desired $LossRate$. If r is not chosen yet, then it can be replaced with $r = r_0 + 1$, and the choice of r_0 is conservative.

Figure 6 shows the trade-off between the bandwidth consumption and the data loss rate (in log scale). In this experiment the space-overhead $\frac{(s+r)}{s}$ is fixed 2, this means $r = s$. Increasing r_0 means more reliability ($LossRate$ decreases exponentially) at the cost of more bandwidth consumption. Note that the bandwidth consumption increases very fast when r_0 is close to r.

For example, to provision a system to have $LossRate < 10^{-20}$ (20 nines of reliability, which is more than many RAID and NAS systems), and peer's $MTTF$ of 1 year, we find the value $r_0 = 10$ using the Equation (3). Then, the bandwidth consumption comes directly from Equation (2).

C. Determining the redundancy (r)

When s and r are defined, provisioning the system is easy and rely on the choice of the best r_0 that matches the resource constraints. However this is not always the case. If the space-overhead is not a problem, the parameter r can be chosen in such way that the bandwidth consumption is optimal. Figure 7 shows an experiment with fixed $s = 16$ and $r_0 = 6$, and increasing values of r. The results show that

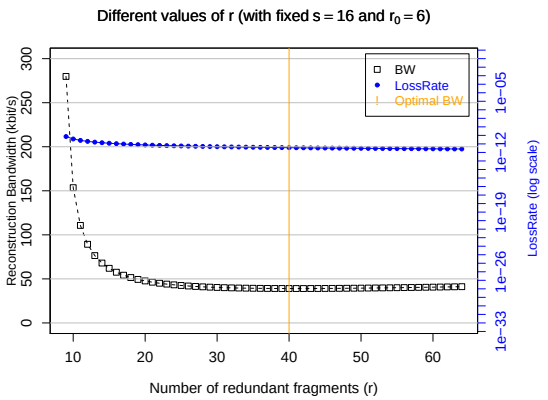

Fig. 7. System with fixed values of s and r_0, and increasing values of r (there is no space-overhead limit).

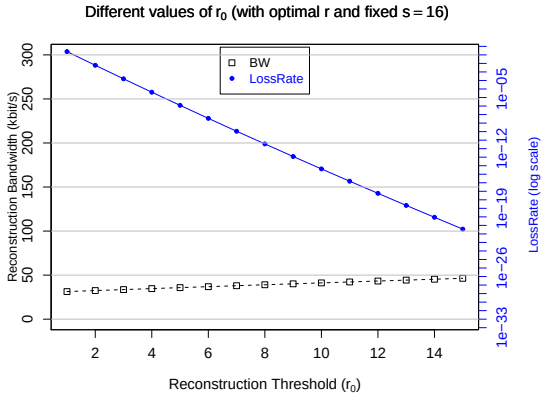

Fig. 8. System with fixed s and increasing values of r_0. The value r is defined by the optimal bandwidth utilisation.

higher values of r decrease slightly the $LossRate$, however, the BW_{avg} follows a parabolic shape, for low values of r (extreme case $r = r_0 + 1$, the eager policy) the bandwidth consumption is very high, then it decays very fast. At a certain point the bandwidth consumption starts to grow with r.

Intuitively, we increase the value of r to delay the repair process because the overhead of the blocks' reconstruction. Mainly, we aim at reducing the fraction of blocks at the last Non-Critical state ($s + r_0 + 1$). This strategy has a strong effect when r is close to r_0 (see Equation (1)) but it decreases slowly when $r - r_0$ is large. However, at a certain point, the cost of having more fragments outweigh the gains of reducing the fraction of blocks at the state $s + r_0 + 1$.

To obtain the best bandwidth consumption for a given s and r_0, it is sufficient to find the derivative of Equation (2) with respect to r. The best r is given by $\frac{\partial BW_{avg}}{\partial r} = 0$, which evaluates to the following equation:

$$r_0 - s - r + (s + r) \cdot \ln\left(\frac{s + r}{s + r_0}\right) = 0 \qquad (4)$$

The term r can then be isolated numerically, but it does not have a nice readable form. In the given example, for $s = 16$ and $r_0 = 8$, the optimal value of r equals 40 (space-overhead of 3.5). In this case, the average bandwidth consumption is 39.1 Kbps per peer, instead of 57.8 Kbps when the space-overhead is 2. Figure 8 shows a system with increasing values of r_0, and r chosen accordingly to Equation (4). The bandwidth consumption is optimal for those values of s and r_0. Note that the bandwidth consumption increases very slowly, while the $LossRate$ decreases exponentially.

VI. Conclusion

In this paper, we analyzed the steady-state of a peer-to-peer storage system based on erasure codes and lazy repair. From a simplified Markov Chain Model we deduced close-form mathematical expressions to estimate the system behavior. The results were focused on the metrics: probability to lose data and bandwidth consumption. We described a methodology to determine the main system parameters, such as the number of initial fragments s, the reconstruction threshold r_0 and the space-overhead defined by $(s + r)/s$. We show that the lazy repair mechanism can be employed to achieve a better utilization of bandwidth for a given reliability, at the cost of additional space usage.

Acknowledgment

This work was partially funded by the ANR projects SPREADS and DIMAGREEN.

References

[1] S. Alouf, A. Dandoush, and P. Nain. Performance analysis of peer-to-peer storage systems. *Internation Teletraffic Congress (ITC), LNCS 4516*, 4516:642–653, 2007.

[2] R. Bhagwan, K. Tati, Y. chung Cheng, S. Savage, and G. M. Voelker. Total recall: System support for automated availability management. In *Proc. of NSDI*, pages 337–350, 2004.

[3] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs. *SIGMETRICS Perform. Eval. Rev.*, 28(1):34–43, 2000.

[4] J.-M. Busca, F. Picconi, and P. Sens. Pastis: A highly-scalable multi-user peer-to-peer file system. In *11th International Euro-Par Conference (Euro-Par'05)*, volume 3648, pages 1173–1182. Springer, 2005.

[5] B.-G. Chun, F. Dabek, A. Haeberlen, E. Sit, H. Weatherspoon, M. F. Kaashoek, J. Kubiatowicz, and R. Morris. Efficient replica maintenance for distributed storage systems. In *Proc. of NSDI*, pages 45–48, 2006.

[6] O. Dalle, F. Giroire, J. Monteiro, and S. Pérennes. Analysis of failure correlation impact on peer-to-peer storage systems. In *Proc. of IEEE P2P*, pages 184–193, Sep 2009.

[7] A. Datta and K. Aberer. Internet-scale storage systems under churn – a study of the steady-state using markov models. In *Proc. of IEEE P2P*, pages 133–144. IEEE Computer Society, 2006.

[8] A. Dimakis, P. Godfrey, M. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. In *Proc. of IEEE INFOCOM*, pages 2000–2008, May 2007.

[9] W. Lin, D. Chiu, and Y. Lee. Erasure code replication revisited. In *Proc. of P2P Computing*, pages 90–97, 2004.

[10] M. Luby, M. Mitzenmacher, M. Shokrollahi, D. Spielman, and V. Stemann. Practical loss-resilient codes. In *Proc. ACM Symp. on Theory of computing*, pages 150–159, 1997.

[11] S. Ramabhadran and J. Pasquale. Analysis of long-running replicated systems. In *Proc. of INFOCOM*, pages 1–9, 2006.

[12] R. Rodrigues and B. Liskov. High availability in dhts: Erasure coding vs. replication. In *Peer-to-Peer Systems IV*, pages 226–239. LNCS, 2005.

[13] A. Rowstron and P. Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *Proc. ACM SOSP*, pages 188–201, 2001.

[14] UbiStorage. http://www.ubistorage.com/. Last accessed: Dec. 2009.

[15] H. Weatherspoon and J. Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In *Proc. of IPTPS*, pages 328–338, 2002.

[16] Z. Wilcox-O'Hearn and B. Warner. Tahoe: the least-authority filesystem. In Y. Kim and W. Yurcik, editors, *ACM StorageSS*, pages 21–26, 2008.

[17] Wuala - secure online storage. http://www.wuala.com/. Last accessed: Dec. 2009.