Performance analysis of centralized versus distributed recovery schemes in P2P storage systems

Abdulhalim Dandoush, Sara Alouf, Philippe Nain

Performance Analysis of Centralized versus Distributed Recovery Schemes in P2P Storage Systems

Abdulhalim Dandoush, Sara Alouf and Philippe Nain
INRIA Sophia Antipolis – B.P. 93 – 06902 Sophia Antipolis, France
{adandous, salouf, nain}@sophia.inria.fr

Abstract

This paper studies the performance of Peer-to-Peer Storage Systems (P2PSS) in terms of data lifetime and availability. Two schemes for recovering lost data are modeled through absorbing Markov chains and their performance are evaluated and compared. The first scheme relies on a centralized controller that can recover multiple losses at once, whereas the second scheme is distributed and recovers one loss at a time. The impact of each system parameter on the performance is evaluated, and guidelines are derived on how to engineer the system and tune its key parameters in order to provide desired lifetime and/or availability of data. We find that, in stable environments such as local area or research laboratory networks where machines are usually highly available, the distributed-repair scheme offers a reliable, scalable and cheap storage/backup solution. This is in contrast with the case of highly dynamic environments, where the distributed-repair scheme is inefficient as long as the storage overhead is kept reasonable. P2PSS with centralized-repair scheme are efficient in any environment but have the disadvantage of relying on a centralized authority. Our analysis also suggests that the use of large size fragments reduces the efficiency of the recovery mechanism.

keywords — peer-to-peer storage systems, recovery process, absorbing continuous-time Markov chain, performance evaluation

1 Introduction

The growth of storage volume, bandwidth, and computational resources for PCs has fundamentally changed the way applications are constructed, and has inspired a new class of storage systems that use distributed peer-to-peer (P2P) infrastructures. Although scalable and economically attractive compared to traditional systems, these storage systems pose many problems such as reliability, confidentiality and availability. To ensure data reliability and availability in such dynamic systems, redundant data is inserted in the system. However, using redundancy mechanisms without repairing lost data is not efficient, as the level of redundancy decreases when peers leave the system. Consequently, P2P storage systems need to compensate the loss of data by continuously storing additional redundant data onto new hosts. Systems may rely on a central authority that reconstructs fragments when necessary; these systems will be referred to as *centralized-recovery systems*. Alternatively, secure agents running on new hosts can reconstruct by themselves the data to be stored on the hosts disks. Such systems will be referred to as *distributed-recovery systems*. Regardless of the recovery mechanism used, two repair policies can be adopted *eager* and *lazy* policy. The description of these recovery schemes and their policies are presented in Section 3.

The aim of this paper is to develop and evaluate mathematical models to characterize fundamental performance metrics (data lifetime and availability) of P2P storage systems. Our contributions are as follows

- Analysis of centralized and distributed recovery mechanisms.
- Proposition of a general model that captures the behavior of both eager and lazy repair policies and both replication-based and erasure code-based systems, and accommodates both temporary and permanent disconnections of peers.
- Numerical investigation using realistic parameters values.
- Guidelines on how to engineer the P2PSS in order to satisfy given requirements.

In the following, Section 2 briefly reviews related work and Section 3 introduces the notation and assumptions used throughout the paper. Sections 4 and 5 are dedicated to the modeling of the centralized- and distributed-recovery mechanism, respectively. In Section 6, we provide some numerical results showing the performance of the centralized and decentralized schemes. Section 7 concludes the paper.

2 Related Work and Background

The literature on the architecture and file system of distributed storage systems is abundant (see [12, 3]; non-exhaustive list), but to the best of our knowledge, there has been no prior work on the modeling of the recovery process in P2PSS. A few studies have developed analytical models with the goal of understanding the trade-offs between the availability and lifetime of the files and the redundancy involved in storing the data. From these we cite [10] whose main purpose of [10] is the analysis of a storage system using replication for data reliability. The work [10] is the closest to ours even though the model and analysis developed therein do not apply for erasure-coded systems (we will see later that our models apply to either replicated or erasure-coded systems).

The authors of [10] develop a Markov chain analysis, then derive an expression for the lifetime of the replicated state and study the impact of bandwidth and storage limits on the system. However – and these are major differences with the work presented here, transient disconnections are not considered in their model, the recovery process is considered to be exponentially distributed for the aim of simplification, and only the distributed-repair scheme is considered. Another contribution of [10] is the analysis of the *All-pairs-ping* data set [13] that reports measures of both uptime and downtime for PlanetLab [9] nodes. The authors found that an exponential distribution is a reasonable fit for both uptime and downtime. This conjecture comes to support one of the key assumptions of the model presented in our paper, namely that "node participation can be modeled by an exponential distribution".

In our previous work [1], simplifying assumptions have been considered while modeling the P2PSS, mainly that the recovery process is exponentially distributed. However, the implications of this assumption differ between replicated and erasure-coded systems. We have found in [4], through a simulation analysis, that the exponential assumption made on the recovery process is usually not met in erasure-coded systems. This assumption will therefore be relaxed in this paper and it will be shown later on that a more precise, more realistic modeling is possible.

3 System Description, Assumptions and Notation

We consider a distributed *storage* system which peers randomly join and leave. The following assumptions on the P2PSS design will be enforced throughout the paper:

- A block of data D is partitioned into s equally sized fragments to which, using erasure codes (e.g. [11]), r redundant fragments are added. The case of replication-based redundancy is equally captured by this notation, after setting $s = 1$ and letting the r redundant fragments be simple replicas of the unique fragment of the block. This notation – and hence our modeling – is general enough to study both replication-based and erasure code-based storage systems.
- Mainly for privacy issues, a peer can store at most one fragment of any data D.
- We assume the system has perfect knowledge of the location of fragments at any given time, e.g. by using a Distributed Hash Table (DHT) or a central authority.
- The system keeps track of only the latest known location of each fragment.
- Over time, a peer can be either *connected* to or *disconnected* from the storage system. At reconnection, a peer may or may not still store its fragments. We denote by p the probability that a peer that reconnects still stores its fragments.
- The number of connected peers at any time is typically much larger than the number of fragments associated with D, i.e., $s+r$. Therefore, we assume that there are always at least $s+r$ connected peers – hereafter referred to as *new* peers – which are ready to receive and store fragments of D.

We refer to as *on-time* (resp. *off-time*) a time-interval during which a peer is always connected (resp. disconnected). During a peer's off-time, the fragments stored on this peer are momentarily unavailable to the users of the storage system. At reconnection, and according to the assumptions above, the fragments stored on this peer will be available

only with probability p (and with probability $1-p$ it is lost). In order to improve data availability and increase the reliability of the storage system, it is therefore crucial to recover from losses by continuously monitoring the system and adding redundancy whenever needed.

We will investigate the performance of two different repair policies: the *eager* and the *lazy* repair policies. In the eager policy, a fragment of D is reconstructed as soon as one fragment has become unavailable due to a peer disconnection. In the lazy policy, the repair is delayed until the number of unavailable fragments reaches a given threshold, denoted k. In the latter case, we must have $k \leq r$ since D is lost if more than r fragments are missing from the storage system. Both repair policies can be represented by the threshold parameter $k \in \{1, 2, \ldots, r\}$, where k can take any value in the set $\{2, \ldots, r\}$ in the lazy policy and $k = 1$ in the eager policy. Any repair policy can be implemented either in a centralized or a distributed way. In the following description, we assume that the system misses k fragments so that lost fragments have to be restored.

In the centralized implementation, a central authority will: (1) download *in parallel* s fragments from the peers which are connected, (2) reconstruct at once all the unavailable fragments, and (3) upload them all *in parallel* onto as many new peers for storage. Step 2 executes in a negligible time compared to the execution time of Steps 1 and 3 and will henceforth be ignored in the modeling. Step 1 (resp. Step 3) execution completes when the last fragment completes being downloaded (resp. uploaded).

In the distributed implementation, a secure agent on one new peer is notified of the identity of *one* out of the k unavailable fragments for it to reconstruct it. Upon notification, the secure agent (1) downloads s fragments of D from the peers which are connected to the storage system, (2) reconstructs the specified fragment and stores it on the peer's disk; (3) the secure agent then discards the s downloaded fragments so as to meet the privacy constraint that only one fragment of a block of data is held by a peer. This operation iterates until less than k fragments are sensed unavailable and stops if the number of missing fragments reaches $k-1$. The recovery of one fragment lasts mainly for the execution time of Step 1; the recovery is completed then as soon as the last fragment (out of s) completes being downloaded.

In both implementations, once a fragment is reconstructed, any other copy of it that "reappears" in the system due to a peer reconnection is simply ignored, as only one location (the newest) of the fragment is recorded in the system. Similarly, if a fragment is unavailable, the system knows of only one disconnected peer that stores the unavailable fragment.

Given the system description, data D can be either *available*, *unavailable* or *lost*. Data D is said to be available if any s fragments out of the $s+r$ fragments can be downloaded by the users of the P2PSS. Data D is said to be unavailable if *less than* s fragments are available for download, however the missing fragments to complete D are located at a peer or a central authority on which a recovery process is ongoing. Data D is said to be lost if there are *less than* s fragments in the system including the fragments involved in a recovery process. We assume that, at time $t = 0$, at least s fragments are available so that the document is initially available.

We now introduce the assumptions considered in our models.

Assumption 1: We assume that the successive durations of on-times (resp. off-times) of a peer are independent and identically distributed (iid) random variables (rvs) with a common exponential distribution function with parameter $\mu > 0$ (resp. $\lambda > 0$); this assumption is in agreement with the analysis in [10]. We further assume that peers behave independently of each other.

Assumption 2: We assume that the successive download (resp. upload) durations of a fragment are iid rvs with a common exponential distribution function with parameter α (resp. β); this assumption is in agreement with the analysis in [4]. Fragments downloads/uploads are not correlated.

A consequence of Assumption 2 is that each of the durations of the centralized and the distributed recovery processes is a rv following a hypo-exponential distribution [7]. Indeed, each of these durations is the summation of independently distributed exponential rvs ($s+k$ in the centralized scheme if k fragments are to be reconstructed, and s in the distributed scheme) having each its own rate. This is a fundamental difference with [10, 1] where the recovery process is assumed to follow an exponential distribution. It is worth mentioning that the simulation analysis of [4] has concluded that the recovery time follows roughly a hypo-exponential distribution. As a consequence, the models presented in this paper are more realistic than those in [10, 1].

We conclude this section by a word on the notation: a subscript "c" (resp. "d") will indicate that we are considering the centralized (resp. distributed) scheme. The notation $\vec{e}_j^{\,i}$ refers to a *row* vector of dimension j whose entries are null except the i-th entry that is equal to 1; the notation $\vec{1}_j$ refers to a *column* vector of dimension j whose each entry is equal to 1. Last, $\mathbb{1}\{A\}$ is the characteristic function of event A.

4 Centralized Repair Systems

We will focus on a single block of data D, and pay only attention to peers storing fragments of this block.

Let $X_c(t)$ and $Y_c(t)$ be two rvs denoting respectively the number of fragments in the system that are available for download and the state of the recovery process. Recall that, when k fragments are to be reconstructed, the recovery process consists of a series of $s+k$ exponential distributions that can be seen as $s+k$ stages. We denote $Y_c(t)=j$ $(j=0,1,\ldots,k-1)$ to express that j exponential rvs have been realized at time t, so that $s+k-j$ are still to go. When the last stage is completed, the recovery process is completed and $Y_c(t)=0$. Given that there could be as much as $s+r$ fragments to be reconstructed, the process $Y_c(t)$ takes value in the set $\{0,1,\ldots,2s+r-1\}$. As for $X_c(t)$, it takes value in the set $\{0,1,\ldots,s+r\}$.

Consider now the joint process $(X_c(t),Y_c(t))$. When $X_c(t)\geq s$, data D is *available*, regardless of $Y_c(t)$. When $X_c(t)<s$ but $X_c(t)+Y_c(t)\geq s$, D is *unavailable*. When $X_c(t)+Y_c(t)<s$, D is *lost*. The latter situation will be modeled by a single state a. Introduce the set

$$\begin{aligned}\mathcal{T}_c := \{\ & \left.\begin{array}{l}(0,s),(0,s+1),\ldots,(0,2s+r-1),\\ (1,s-1),(1,s),\ldots,(1,2s+r-2),\\ \ldots,\end{array}\right\} && D \text{ is unavailable}\\ & \left.\begin{array}{l}(s,0),(s,1),\ldots,(s,s+r-1),\\ (s+1,0),(s+1,1),\ldots,(s+1,s+r-2),\\ \ldots,(s+r-1,0),(s+r-1,1),\ldots,(s+r-1,s),\\ (s+r,0)\ \}\end{array}\right\} && D \text{ is available}\\ |\mathcal{T}_c| &= (s+r)^2-r(r-1)/2+1.\end{aligned}$$

Thanks to the assumptions made in Section 3, it is easily seen that the two-dimensional process $\{(X_c(t),Y_c(t)),\ t\geq 0\}$ is an absorbing homogeneous Continuous-Time Markov Chain (CTMC) with transient states the elements of $\mathcal{T}_c$ and with a single absorbing state a representing the situation when D is lost. Without loss of generality, we assume that $X_c(0)\geq s$. The infinitesimal generator has the following canonical form

$$\begin{array}{c} \\ \mathcal{T}_c \\ a \end{array}\begin{array}{c} \begin{array}{cc}\mathcal{T}_c & a\end{array} \\ \left(\begin{array}{c|c}\vec{Q}_c & \vec{R}_c \\ \hline \vec{0} & 0\end{array}\right)\end{array}$$

where $\vec{R}_c$ is a non-zero column vector of size $|\mathcal{T}_c|$, and $\vec{Q}_c$ is $|\mathcal{T}_c|$-by-$|\mathcal{T}_c|$ matrix. The elements of $\vec{R}_c$ are the transition rates between the transient states $(i,j)\in\mathcal{T}_c$ and the absorbing state a, namely, $r_c(i,j)=(s-j)\mu$, for $i=1,\ldots,s$, and $j=s-i,\ldots,s-1$. The elements of $\vec{R}_c$ are lexicographically ordered alike the order in $\mathcal{T}_c$. The diagonal elements of $\vec{Q}_c$ are each the total transition rate out of the corresponding transient state. The other elements of $\vec{Q}_c$ are the transition rates between each pair of transient states. The non-zero elements of $\vec{Q}_c$ are:

$$q_c((i,j),(i-1,j)) = \begin{cases} i\mu, & \text{for } i=1,\ldots,s, j=s,\ldots,2s+r-1-i;\\ & \text{or } i=s+1,\ldots,s+r-1,\\ & \quad j=0,\ldots,2s+r-1-i;\\ & \text{or } i=s+r, j=0;\\ (i+j-s)\mu, & \text{for } i=2,\ldots,s, j=s+1-i,\ldots,s-1.\end{cases}$$

$$q_c((i,j),(i,j+1)) = \begin{cases} (s-j)\alpha, & \text{for } i=s,\ldots,s+r-k, j=0;\\ & \text{or } i=1,\ldots,s-1, j=s-i,\ldots,s-1;\\ & \text{or } i=s,\ldots,s+r-1, j=1,\ldots,s-1;\\ (2s+r-i-j)\beta, & \text{for } i=0,\ldots,s+r-2,\\ & \quad j=s,\ldots,2s+r-2-i.\end{cases}$$

$$\begin{aligned} q_c((i,2s+r-1-i),(s+r,0)) &= \beta, && \text{for } i=0,\ldots,s+r-1.\\ q_c((i,j),(i+1,j)) &= (s+r-i)\lambda p, && \text{for } i=1,\ldots,s, j=s-i,\ldots,s-1;\\ & && \text{or } i=s+1,\ldots,s+r-2, j=0,\ldots,s-1.\\ q_c((s+r-1,j),(s+r,0)) &= \lambda p, && \text{for } j=0,\ldots,s-1.\\ q_c((i,j),(i,j)) &= -r_c(i,j)-\textstyle\sum_{(i',j')\in\mathcal{T}_c-\{(i,j)\}} q_c((i,j),(i',j')), && \text{for } (i,j)\in\mathcal{T}_c.\end{aligned}$$

Note that $\vec{Q}_c$ is not an infinitesimal generator since entries in some rows do not sum up to 0. For illustration purposes, we depict in Fig. 1 an example of the absorbing CTMC with its non-zero transition rates when $s=2$, $r=2$, and $k=2$.

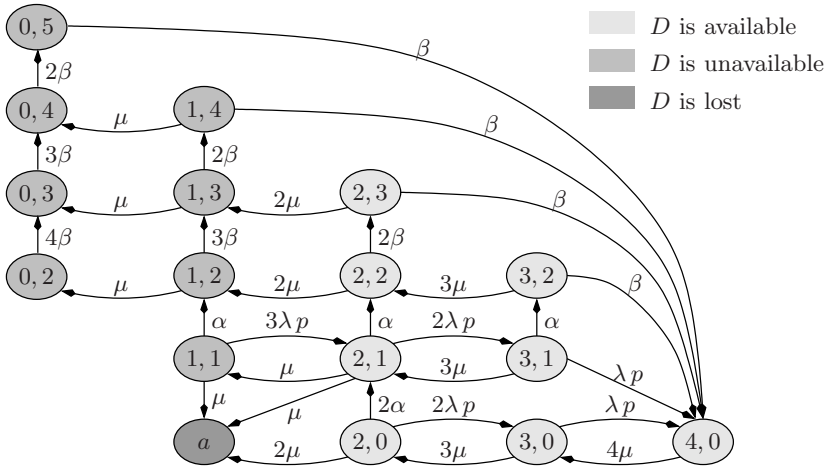

Figure 1: The Markov chain $\{(X_c(t), Y_c(t)),\, t \geq 0\}$ when $s = 2$, $r = 2$, $k = 2$.

4.1 Data Lifetime

This section is devoted to the analysis of the lifetime of D. Let $T_c(i,j) := \inf\{t > 0 : (X_c(t), Y_c(t)) = a | (X_c(0), Y_c(0)) = (i,j)\}$ be the time until absorption in state a, or equivalently the time until D is lost, given that the initial state of D is (i,j). In the following, $T_c(i,j)$ will be referred to as the *conditional block lifetime*. We are interested in $P(T_c(i,j) \leq x)$ and $\mathrm{E}[T_c(i,j)]$, respectively the probability distribution of the conditional block lifetime and its expectation, given that $(X_c(0), Y_c(0)) = (i,j) \in \mathcal{T}_c$. From the theory of absorbing Markov chains, we know that (e.g. [8, Lemma 2.2])

$$P(T_c(i,j) \leq x) = 1 - \vec{e}_{|\mathcal{T}_c|}^{\,\mathrm{ind}(i,j)} \cdot \exp\left(x\vec{Q}_c\right) \cdot \vec{1}_{|\mathcal{T}_c|}, \quad x > 0,\ (i,j) \in \mathcal{T}_c \tag{1}$$

where $\mathrm{ind}(i,j)$ refers to the index of the state $(i,j) \in \mathcal{T}_c$ in the matrix $\vec{Q}_c$. Recall that the elements of $\vec{Q}_c$ are numbered according to the lexicographic order. Definitions of vectors $\vec{e}_i^{\,j}$ and $\vec{1}_i$ are given at the end of Section 3. Observe that the term $\vec{e}_{|\mathcal{T}_c|}^{\,\mathrm{ind}(i,j)} \cdot \exp\left(x\vec{Q}_c\right) \cdot \vec{1}_{|\mathcal{T}_c|}$ in the r.h.s. of (1) is nothing but the summation of all $|\mathcal{T}_c|$ elements in row $\mathrm{ind}(i,j)$ of matrix $\exp\left(x\vec{Q}_c\right)$.

We know from [8, p. 46] that the expected time until absorption can be written as

$$\mathrm{E}\left[T_c(i,j)\right] = -\vec{e}_{|\mathcal{T}_c|}^{\,\mathrm{ind}(i,j)} \cdot \left(\vec{Q}_c\right)^{-1} \cdot \vec{1}_{|\mathcal{T}_c|}, \quad (i,j) \in \mathcal{T}_c, \tag{2}$$

where the existence of $\left(\vec{Q}_c\right)^{-1}$ is a consequence of the fact that all states in $\mathcal{T}_c$ are transient [8, p. 45]. Inverting $\vec{Q}_c$ analytically can rapidly become cumbersome as s or r increases. We will instead perform numerical computations as reported in Section 6. Consider now

$$T_c((i,j),(i',j')) := \int_0^{T_c(i,j)} \mathbb{1}\{(X_c(t), Y_c(t)) = (i',j')\}\, dt$$

that is the total time spent by the CTMC in transient state (i',j') given that $\{X_c(0), Y_c(0)\} = (i,j)$. It can also be shown that [5, p. 419]

$$\mathrm{E}\left[T_c((i,j),(i',j'))\right] = -\vec{e}_{|\mathcal{T}_c|}^{\,\mathrm{ind}(i,j)} \cdot \left(\vec{Q}_c\right)^{-1} \cdot {}^t\vec{e}_{|\mathcal{T}_c|}^{\,\mathrm{ind}(i',j')}, \quad (i,j),(i',j') \in \mathcal{T}_c, \tag{3}$$

where ${}^t\vec{y}$ denotes the transpose of a given vector $\vec{y}$. In other words, the expectation $\mathrm{E}\left[T_c((i,j),(i',j'))\right]$ is the entry of matrix $\left(-\vec{Q}_c\right)^{-1}$ at row $\mathrm{ind}(i,j)$ and column $\mathrm{ind}(i',j')$.

4.2 Data Availability

In this section we introduce different metrics to quantify the availability of D. We are interested in the fraction of time spent by the CTMC in any given state (i', j') before absorption. However, this quantity is difficult to find in closed-form. Therefore, we resort to using the following approximation

$$\mathrm{E}\left[\frac{T_c((i,j),(i',j'))}{T_c(i,j)}\right] \approx \frac{\mathrm{E}[T_c((i,j),(i',j'))]}{\mathrm{E}[T_c(i,j)]}. \tag{4}$$

Here, (i, j) is the state of D at $t = 0$. This approximation have been validated through simulations, as shown later in Section 6. With this approximation in mind, we introduce two availability metrics: the first can be interpreted as the expected number of fragments of D that are in the system during the lifetime of D; the second can be interpreted as the fraction of time when at least m fragments are in the system during the lifetime of D. More formally, given that $(X_c(0), Y_c(0)) = (i, j) \in \mathcal{T}_c$, we define

$$M_{c,1}(i,j) := \sum_{(i',j')\in\mathcal{T}_c} i' \frac{\mathrm{E}[T_c((i,j),(i',j'))]}{\mathrm{E}[T_c(i,j)]}, \tag{5}$$

$$M_{c,2}((i,j),m) := \sum_{(i',j')\in\mathcal{T}_c, i'\geq m} \frac{\mathrm{E}[T_c((i,j),(i',j'))]}{\mathrm{E}[T_c(i,j)]}. \tag{6}$$

5 Distributed Repair Systems

In this section, we model P2P storage systems that implement a distributed recovery mechanism. According to the description and assumptions listed in Section 3, the state of data D can be modeled by an absorbing Markov chain $\{(X_d(t), Y_d(t)) : t \geq 0\}$, where $X_d(t)$ and $Y_d(t)$ denote respectively the number of fragments in the system that are available for download and the state of the recovery process. Unlike the centralized scheme, the distributed scheme repairs fragments only one at a time. Therefore, $X_d(t)$ takes value in the set $\{s-1, s, \ldots, s+r\}$. There are only s stages in the recovery process that correspond each to an exponential with it own rate (i.e., $Y_d(t) \in \{0, 1, \ldots, s-1\}$). As in Section 4, D is available when $X_d(t) \geq s$, unavailable when $X_d(t) < s$ but $X_d(t) + Y_d(t) \geq s$, and lost otherwise (situation modeled by a single absorbing state a). The set of transient states $\mathcal{T}_d$ is

$$\mathcal{T}_d := \left\{ \begin{array}{l} \left.\begin{array}{l} (s-1,1),(s-1,2),\ldots,(s-1,s-1), \end{array}\right\} D \text{ is unavailable} \\ \left.\begin{array}{l} (s,0),(s,1),\ldots,(s,s-1), \\ (s+1,0),(s+1,1),\ldots,(s+1,s-1),\ldots, \\ (s+r-1,0),(s+r-1,1),\ldots,(s+r-1,s-1), \\ (s+r,0) \end{array}\right\} D \text{ is available} \end{array}\right\}$$

$$|\mathcal{T}_d| = s(r+1).$$

The analysis of the absorbing Markov chain $\{(X_d(t), Y_d(t)) : t \geq 0\}$ that takes value in $\mathcal{T}_d \cup \{a\}$ is very similar to the analysis in Section 4, we will then only sketch it. In particular, $\vec{R}_d$ and $\vec{Q}_d$ have similar definitions as $\vec{R}_c$ and $\vec{Q}_c$ after replacing the subscript "c" with the subscript "d" whenever needed. The non-zero elements of $\vec{R}_d$ and $\vec{Q}_d$ are as follows

$$r_d(s-1,j) = (s-1)\mu, \text{ for } j = 1,\ldots,s-1; \quad r_d(s,j) = (s-j)\mu, \text{ for } j = 0,\ldots,s-1.$$

$$q_d((i,j),(i-1,j)) = \begin{cases} j\mu, & \text{for} \quad i = s, j = 1,\ldots,s-1; \\ i\mu, & \text{for} \quad i = s+1,\ldots,s+r-1, j = 0,\ldots,s-1; \\ & \text{or} \quad i = s+r, j = 0. \end{cases}$$

$$q_d((i,j),(i,j+1)) = (s-j)\alpha, \quad \text{for} \quad i = s,\ldots,s+r-k, j = 0; \quad \text{or} \quad i = s-1,\ldots,s+r-1, j = 1,\ldots,s-2.$$

$$q_d((i,s-1),(i+1,0)) = \alpha, \quad \text{for} \quad i = s-1,\ldots,s+r-2.$$

$$q_d((i,j),(i+1,j)) = (s+r-i)\lambda p, \quad \text{for} \quad i = s-1, j = 1,\ldots,s-1; \quad \text{or} \quad i = s,\ldots,s+r-2, j = 0,\ldots,s-1.$$

$$q_d((s+r-1,j),(s+r,0)) = \lambda p + \mathbb{1}\{j = s-1\}\alpha, \quad \text{for} \quad j = 0,\ldots,s-1.$$

$$q_d((i,j),(i,j)) = -r_d(i,j) - \textstyle\sum_{(i',j')\in\mathcal{T}_d-\{(i,j)\}} q_d((i,j),(i',j')), \quad \text{for } (i,j) \in \mathcal{T}_d.$$

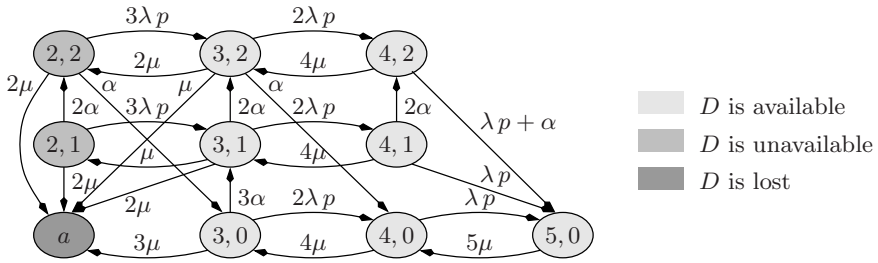

Figure 2: The Markov chain $\{(X_d(t), Y_d(t)),\, t \geq 0\}$ when $s = 3$, $r = 2$, and $k = 2$.

For illustration purposes, we depict in Fig. 2 an example of the absorbing CTMC with its non-zero transition rates when $s = 3$, $r = 2$, and $k = 2$.

We can now derive closed-form expressions for the distribution of the conditional block lifetime, its expectation, and the two availability metrics, as was done in Section 4, by simply replacing in (1), (2), (3), (5) and (6) the subscript "c" with the subscript "d". Alike for the centralized case, we will perform numerical computations as it is not tractable to explicitly invert $\vec{Q}_d$.

6 Numerical Results

In this section, we first validate the approximation made in (4) which has been made to compute the two availability metrics, then proceed with the presentations of the numerical computations. Throughout this section, we consider two sets of parameters that correspond each to a particular context. In the "PlanetLab" context, we set $1/\lambda = 61$ hours and $1/\mu = 181$ hours according to [10], $p = 0.3$ to reflect that disconnections are most likely due to software or hardware problems, and we vary the redundancy r from 1 until s. In the "Internet" context, peers churn rate is much higher [2] (namely, hosts join and leave the system 6.4 times a day on average), which led us to set $1/\lambda = 1$ hours, $1/\mu = 3$ hours, $p = 0.7$, and to vary r from 1 to $2s$. In both contexts, we let $s = 8$, $k = 1, \ldots, r$. We assume the peers' upload capacity that is dedicated to a single connection to be 10kbps (cf. [6]), and the total upload capacity of the central authority to be 400kbps. Hence, considering fragments of size 1MB, we obtain $1/\alpha = 838.8608$ seconds, $1/\beta = 20.97152 \times s$ seconds.

Validation of (4)**.** To this end, we have simulated the CTMC $\{X_d(t), Y_d(t) : t \geq 0\}$ in the "Internet" context but have varied r from 1 to 4, yielding a total of 10 different simulation scenarios. We estimate the left-hand side of (4) by averaging the corresponding simulated results. In order to obtain a maximum estimation error of about 1% with 97% confidence interval, we need to average over 150 sampled values. Hence, each simulation is repeated 150 times. In each simulation, we have a total of $|\mathcal{T}_d| = 8(r+1)$ instances of (4), according to the possible choices of the initial state, yielding a total of $\sum_{r=1}^{4} 8(r+1)r = 320$ different instances. For each instance, we compute the relative error between the estimation of the left-hand side of (4) (the "correct" value) and the right-hand side of (4) (the approximate value, computed analytically using (2)-(3)).

We have found only 10% of the values that are larger than 0.9×10^{-3} and, most importantly, the maximum value of the relative error is 0.0028. *We conclude that the approximation* (4) *is very good and will definitely not imperil the correctness of any computation based on it.*

The empirical complementary cumulative distribution function of the relative error is displayed in Fig. 3.

Performance analysis. We have solved numerically (1), (2), (3), (5) and (6) given that all $s + r$ fragments of D are initially available, considering either Internet or PlanetLab context, and either the centralized or distributed recovery scheme. Results are reported partially in Table 1. It appears that, whichever the scenario or the recovery mechanism considered, the expected data lifetime increases roughly exponentially with r and decreases with an increasing k. Regardless of the context considered, the distributed scheme yields a significantly smaller expected data lifetime than the centralized scheme, especially when the storage overhead, r/s, is high; cf. columns 3-4 in Table 1. The difference in performance is more pronounced in the Internet context. Regarding the expected number of available fragments, we again observe that the distributed scheme is less efficient than the centralized one. Observe how the performance deteriorates as peer churn becomes more important: compare for instance in Table 1 rows 4 vs. 16, and 7 vs. 20 (these correspond to the same storage overhead and the same value of k). This is particularly true for the distributed recovery mechanism. We conclude that *when peers churn rate is high, only the centralized repair scheme*

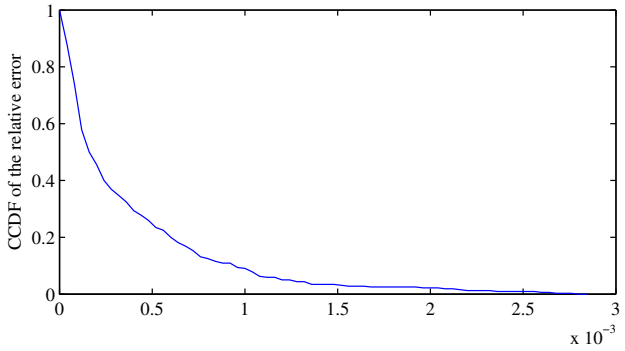

Figure 3: The CCDF of the relative error induced by the approximation (4).

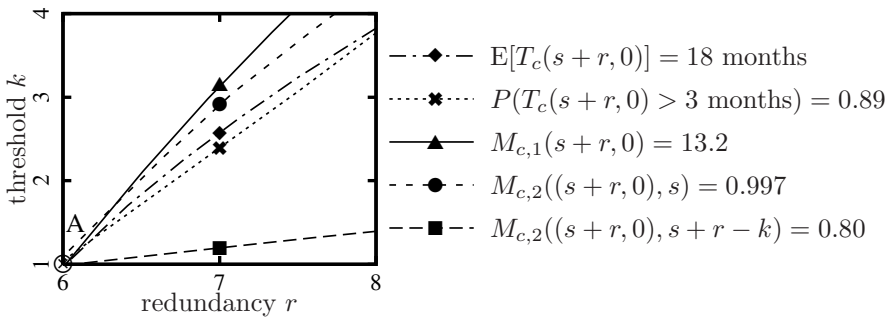

Figure 4: Contour lines of performance metrics (PlanetLab context, centralized repair).

can be efficient should the storage overhead be kept within a reasonable value (that is $r/s \leq 2$). As the distributed repair scheme is more scalable than the centralized one, it will be a good implementation choice in large networks where hosts have a good availability.

Setting the system's key parameters. We illustrate now how our models can be used to set the system parameters r and k such that predefined requirements on data lifetime and availability are fulfilled. We assume the recovery mechanism is centralized and the context is similar to PlanetLab. We have picked one to two contour lines of each of the performance metrics studied in this paper and report them in Fig. 4. Consider point A which corresponds to $r = 6$ and $k = 1$ (recall $s = 8$). Selecting this point as the operating point of the P2PSS ensures (roughly) the following: given that each data is initiated with $s + r$ available fragments, then (i) the expected data lifetime is 18 months; (ii) only 11% of the stored data would be lost after 3 months; (iii) as long as D is not lost, 13 fragments of D are expected to be in the system; (iv) during 99.7% of its lifetime, D is available for download; and (v) during 80% of the lifetime of D, at least $s + r - k = 13$ fragments of D are available for download in the system. Observe that the storage overhead, r/s, is equal to 0.75.

Impact of the size of fragments. Given the size of data D, a larger size of fragments translates into a smaller s. We have computed all pairs (r, k) with $s = 8$ and $s = 16$ that ensure $P(T_c(s + r, 0) > 3 \text{ months}) = 0.89$ in the PlanetLab context, i.e., only 11% of the total data would be lost after 3 months. In particular, operating points $r = 6$ and $k = 1$ with $s = 8$, and $r = 12$ and $k = 7$ with $s = 16$ satisfy the above requirement, and additionally yield the same storage overhead (namely, 0.75). But, and this is important, the former point invokes the recovery process much more often (and potentially unnecessarily) than the latter point, suggesting that *large fragments size reduces the efficiency of the recovery mechanism.* This observation should be moderated by the fact that fragments size when $s = 8$ is twice their size when $s = 16$, yielding a different bandwidth usage per recovery. We currently cannot say how does the bandwidth usage per recovery vary with the size of fragments. However, we know for sure that its effect will not be the same in both centralized and distributed schemes because of the additional upload stages in the centralized implementation.

Table 1: Expected lifetime and first availability metric

Internet context		$\mathrm{E}[T(s+r,0)]$ (in days)		$M_1(s+r,0)$	
	$s=8$	cent. repair	dist. repair	cent. repair	dist. repair
$k=1$	$r/s=1/2$	0.14	0.08	9.91	8.99
	$r/s=1$	0.85	0.35	11.72	10.58
	$r/s=3/2$	14.49	2.25	14.46	12.40
	$r/s=2$	105.80	23.61	16.55	14.73
$k=4$	$r/s=1$	0.71	0.33	11.66	9.67
	$r/s=3/2$	12.38	2.23	14.45	11.50
	$r/s=2$	91.59	23.58	16.48	13.83
$k=8$	$r/s=2$	64.16	22.31	16.30	12.61
PlanetLab context		$\mathrm{E}[T(s+r,0)]$ (in months)		$M_1(s+r,0)$	
	$s=8$	cent. repair	dist. repair	cent. repair	dist. repair
$k=1$	$r/s=1/4$	0.32	0.11	7.81	8.04
	$r/s=1/2$	2.15	1.05	11.01	8.68
	$r/s=3/4$	17.12	7.61	13.18	9.80
	$r/s=1$	262.16	46.24	15.11	12.12
$k=2$	$r/s=1/2$	0.81	0.37	10.34	8.19
	$r/s=3/4$	6.95	3.20	12.76	9.25
	$r/s=1$	110.03	23.34	14.72	11.37
$k=4$	$r/s=1$	13.33	4.34	13.77	9.81

A careful analysis of this issue is the objective of ongoing research.

7 Conclusion

We have proposed analytical models for evaluating the performance of two approaches for recovering lost data in distributed storage systems. We have analyzed the lifetime and the availability of data achieved by both centralized- and distributed-repair systems through Markovian analysis considering realistic assumptions. Numerical computations have been undertaken to illustrate several issues on the performance. We conclude that, using our theoretical framework, it is easy to tune and optimize the system parameters for fulfilling predefined requirements.

References

[1] S. Alouf, A. Dandoush, and P. Nain. Performance analysis of peer-to-peer storage systems. In *Proc. of 20th ITC*, volume 4516 of *Lecture Notes in Computer Science*, pages 642–653, Ottawa, Canada, June 2007.

[2] R. Bhagwan, S. Savage, and G. Voelker. Understanding availability. In *Proc. of 2nd IPTPS*, Berkeley, California, February 2003.

[3] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G.M. Voelker. Total Recall: System support for automated availability management. In *Proc. of ACM/USENIX NSDI '04*, pages 337–350, San Francisco, California, March 2004.

[4] A. Dandoush, S. Alouf, and P. Nain. Simulation analysis of download and recovery processes in P2P storage systems. Technical Report RR-6858, INRIA Sophia Antipolis, February 2009.

[5] C. Grinstead and J. Laurie Snell. *Introduction to Probability*. American Mathematical Society, 1997.

[6] A. Guha, N. Daswani, and R. Jain. An experimental study of the skype peer-to-peer VoIP system. In *Proc. of 5th IPTPS*, Santa Barbara, California, February 2006.

[7] P. Harrison and S. Zertal. Queueing models of RAID systems with maxima of waiting times. *Performance Evaluation Journal*, 64(7-8):664–689, August 2007.

[8] M.F. Neuts. *Matrix Geometric Solutions in Stochastic Models. An Algorithmic Approach.* John Hopkins University Press, Baltimore, 1981.

[9] PlanetLab. An open platform for developing, deploying, and accessing planetary-scale services. `http://www.planet-lab.org/`, 2007.

[10] S. Ramabhadran and J. Pasquale. Analysis of long-running replicated systems. In *Proc. of IEEE Infocom '06*, Barcelona, Spain, April 2006.

[11] I.S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of SIAM*, 8(2):300–304, June 1960.

[12] A. Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Proc. of ACM SOSP '01*, pages 188–201, Banff, Canada, October 2001.

[13] J. Stribling. PlanetLab - All Pairs Pings. `http://pdos.csail.mit.edu/~strib/pl_app`, 2005.