HAL
open science

Performance analysis of peer-to-peer storage systems

Sara Alouf, Abdulhalim Dandoush, Philippe Nain

To cite this version:

Sara Alouf, Abdulhalim Dandoush, Philippe Nain. Performance analysis of peer-to-peer storage systems. 20th international teletraffic conference on Managing traffic performance in converged networks (ITC '07), Jun 2007, Ottawa, Canada. pp.642-653, 10.1007/978-3-540-72990-7_57 . hal-00641279

HAL Id: hal-00641279

https://inria.hal.science/hal-00641279v1

Submitted on 11 Jul 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Performance Analysis of Peer-to-Peer Storage Systems

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

Abstract

This paper evaluates the performance of two schemes for recovering lost data in a peer-to-peer (P2P) storage systems. The first scheme is centralized and relies on a server that recovers multiple losses at once, whereas the second one is distributed. By representing the state of each scheme by an absorbing Markov chain, we are able to compute their performance in terms of the delivered data lifetime and data availability. Numerical computations are provided to better illustrate the impact of each system parameter on the performance. Depending on the context considered, we provide guidelines on how to tune the system parameters in order to provide a desired data lifetime.

keywords — Peer-to-Peer systems, performance evaluation, absorbing Markov chain, mean-field approximation

1 Introduction

Traditional storage solutions rely on robust dedicated servers and magnetic tapes on which data are stored. These equipments are reliable, but expensive. The growth of storage volume, bandwidth, and computational resources 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. Some of the recent efforts for building highly available storage system based on the P2P paradigm are Intermemory [6], Freenet [3], OceanStore [13], CFS [4], PAST [16], Farsite [5] and Total Recall [1]. Although inexpensive compared to traditional systems, these storage systems pose many problems of reliability, confidentiality, availability, routing, etc.

In a P2P network, peers are free to leave and join the system at any time. As a result of the intermittent availability of the peers, ensuring high availability of the stored data is an interesting and challenging problem. To ensure data reliability, redundant data is inserted in the system. Redundancy can be achieved either by replication or by using erasure codes. For the same amount of redundancy, erasure codes provide higher availability of data than replication [18].

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 centralized instance that reconstructs fragments when necessary; these systems will be referred to as *centralized-recovery systems*. Alternatively, secure agents running on new peers can reconstruct by themselves the data to be stored on the peers disks. Such systems will be referred to as *distributed-recovery systems*. A centralized server can recover at once multiple losses of the same document. This is not possible in the distributed case where each new peer thanks to its secure agent recovers only one loss per document.

Regardless of the recovery mechanism used, two repair policies can be adopted. In the *eager* policy, when the system detects that one host has left the system, it immediately repairs the diminished redundancy by inserting a new peer hosting the recovered data. Using this policy, data only becomes unavailable when hosts fail more quickly than they can be detected and repaired. This policy is simple but makes no distinction between permanent departures that require repair, and transient disconnections that do not. An alternative is to defer the repair and to use additional redundancy to mask and to tolerate host departures for an extended period. This approach is called *lazy* repair because the explicit goal is to delay repair work for as long as possible.

In this paper, we aim at developing mathematical models to characterize fundamental performance metrics (lifetime and availability – see next paragraph) of P2P storage systems using erasure codes. We are interested in evaluating the centralized- and distributed-recovery mechanisms discussed earlier, when either eager or lazy repair policy is enforced. We will focus our study on the quality of service delivered to each block of data. We aim at addressing fundamental design issues such as: *how to tune the system parameters so as to maximize data lifetime while keeping a low storage overhead?*

The *lifetime* of data in the P2P system is a random variable; we will investigate its distribution function. *Data availability* metrics refer to the amount of redundant fragments. We will consider two such metrics: the expected number of available redundant fragments, and the fraction of time during which the number of available redundant fragment exceeds a given threshold. For each implementation (centralized/distributed) we will derive these metrics in closed-form through a Markovian analysis.

In the following, Sect. 2 briefly reviews related work and Sect. 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. In Sect. 6, we provide numerical results showing the performance of the centralized and decentralized schemes, under the eager or the lazy policy. We conclude the paper in Sect. 7.

2 Related Work

There is an abundant literature on the architecture and file system of distributed storage systems (see [6, 13, 4, 16, 5, 1]; non-exhaustive list) but only a few studies have developed analytical models of distributed storage systems to understand the trade-offs between the availability of the files and the redundancy involved in storing the data.

In [18], Weatherspoon and Kubiatowicz characterize the availability and durability gains provided by an erasure-resilient system. They quantitatively compare replication-based and erasure-coded systems. They show that erasure codes use an order of magnitude less bandwidth and storage than replication for systems with similar durability. Utard and Vernois perform another comparison between the full replication mechanism and erasure codes through a simple stochastic model for node behavior [17]. They observe that simple replication schemes may be more efficient than erasure codes in presence of low peers availability. In [10], Lin, Chiu and Lee focus on erasure codes analysis under different scenarios, according to two key parameters: the peer availability level and the storage overhead. Blake and Rodrigues argue in [2] that the cost of dynamic membership makes the cooperative storage infeasible in transiently available peer-to-peer environments. In other words, when redundancy, data scale, and dynamics are all high, the needed cross-system bandwidth is unreasonable when clients desire to download files during a reasonable time. Last, Ramabhadran and Pasquale develop in [14] a Markov chain analysis of a storage system using full replication for data reliability, and a distributed recovery scheme. They derive an expression for the lifetime of the replicated state and study the impact of bandwidth and storage limits on the system.

3 System Description and Notation

We consider a distributed storage system in which peers randomly join and leave the system. Upon a peer disconnection, all data stored on this peer is no longer available to the users of the storage system and is considered to be lost. In order to improve data availability it is therefore crucial to add redundancy to the system.

In this paper, we consider a single block of data D, divided into s equally sized fragments to which, using erasure codes (e.g. [15]), r redundant fragments are added. These $s + r$ fragments are stored over $s + r$ different peers. Data D is said to be *available* if any s fragments out of the $s + r$ fragments are available and *lost* otherwise. We assume that at least s fragments are available at time $t = 0$. Note that when $s = 1$ the r redundant fragments will simply be replicas of the unique fragment of the block; replication is therefore a special case of erasure codes.

Over time, a peer can be either *connected* to or *disconnected* from the storage system. At reconnection, a peer may still or may not store one fragment. Data stored on a connected peer is available at once and can be used to reconstruct a block of data. We refer to as *on-time* (resp. *off-time*) a time-interval during which a peer is always connected (resp. disconnected). Typically, the number of connected peers at any time in a storage system is much larger than the number of fragments associated with a given data D. Therefore, we assume that there are always at least r connected peers – hereafter referred to as *new* peers – which are ready to store fragments of D. For security issues, a peer may store at most one fragment.

We assume that the successive durations of on-times (resp. off-times) of a peer form a sequence of independent and identically distributed (iid) random variables (rvs), with an exponential distribution with rate $\alpha_1 > 0$ (resp. $\alpha_2 > 0$). We further assume that peers behave independently of each other, which implies that on-time and off-time sequences associated with any set of peers are statistically independent. We denote by p the probability that a peer that reconnects still stores one fragment and that this fragment is different from all other fragments available in the system.

As discussed in Sect. 1 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 triggered only when the number of unavailable fragments reaches a given threshold $k \geq 1$. Note that $k \leq r$ since D is lost if more than r fragments are not available in the storage system at a given time. Both repair policies can be represented by a threshold parameter $k \in \{1, 2, \ldots, r\}$, where $k = 1$ in the eager policy, and where k can take any value in the set $\{2, \ldots, r\}$ in the lazy policy.

Let us now describe the fragment recovery mechanism. As mentioned in Sect. 1, we will consider two implementations of the eager and lazy recovery mechanisms, a *centralized* and a *(partially) distributed* implementation.

Assume that $k \leq r$ fragments are no longer available due to peer disconnections, triggering the recovery mechanism. In the centralized implementation, a central authority will: (i) download s fragments from the peers which are connected, (ii)

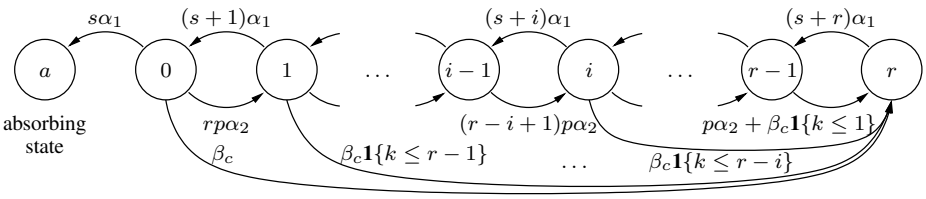


Figure 1: Transition rates of the absorbing Markov chain $\{X_c(t), t \geq 0\}$.

reconstruct at once the k unavailable fragments, and (iii) transmit each of them to a new peer for storage. We will assume that the total time required to perform these tasks is exponentially distributed with rate $\beta_c(k) > 0$ and that successive recoveries are statistically independent.

In the distributed implementation, a secure agent on *one* new peer is notified of the identity of the k unavailable fragments. Upon notification, it downloads s fragments of D from the peers which are connected, reconstructs *one* out of the k unavailable fragments and stores it on its disk; the s downloaded fragments are then discarded so as to meet the security constraint that only one fragment of a block of data is held by a peer. We will assume that the total time required to perform the download, reconstruct and store a new fragment follows an exponential distribution with rate $\beta_d > 0$; we assume that each recovery is independent of prior recoveries.

The exponential distributions have mainly been made for the sake of mathematical tractability. We however believe that these are reasonable assumptions due to the unpredictable nature of the node dynamics and of the variability of network delays.

We conclude this section by a word on the notation: a subscript/superscript "c" (resp. "d") will indicate that we are considering the centralized (resp. distributed) scheme.

4 Centralized Repair Systems

In this section, we address the performance analysis of the centralized implementation of the P2P storage system, as described in Sect. 3. We will focus on a single block of data and we will only pay attention to peers storing fragments of this block.

Let $X_c(t)$ be a $\{a, 0, 1, \ldots, r\}$-valued rv, where $X_c(t) = i \in \mathcal{T} := \{0, 1, \ldots, r\}$ indicates that $s + i$ fragments are available at time t, and $X_c(t) = a$ indicates that less than s fragments are available at time t. We assume that $X_c(0) \in \mathcal{T}$ so as to reflect the assumption that at least s fragments are available at $t = 0$. Thanks to the assumptions made in Sect. 3, it is easily seen that $\vec{X}_c := \{X_c(t),\, t \geq 0\}$ is an absorbing homogeneous Continuous-Time Markov Chain (CTMC) with transient states $0, 1, \ldots, r$ and with a single absorbing state a representing the situation when the block of data is lost. Non-zero transition rates of $\{X_c(t),\, t \geq 0\}$ are shown in Fig. 1.

4.1 Data Lifetime

This section is devoted to the analysis of the data lifetime. Let $T_c(i) := \inf\{t \geq 0 : X_c(t) = a\}$ be the time until absorption in state a starting from $X_c(0) = i$, or equivalently the time at which the block of data is lost. In the following, $T_c(i)$ will be referred to as the *conditional block lifetime*. We are interested in $P(T_c(i) < x)$, the probability distribution of the block lifetime given that $X_c(0) = i$ for $i \in \mathcal{T}$, and the expected time spent by the absorbing Markov chain in transient state j, given that $X_c(0) = i$.

Let $\vec{Q}_c = [q_c(i,j)]_{0 \leq i,j \leq r}$ be a matrix, where for any $i, j \in \mathcal{T}$, $i \neq j$, $q_c(i,j)$ gives the transition rate of the Markov chain $\vec{X}_c$ from transient state i to transient state j, and $-q_c(i,i)$ is the total transition rate out of state i. Non-zero entries of $\vec{Q}_c$ are

$$\begin{array}{lcll} q_c(i,i-1) & = & c_i \ , & i = 1, 2, \ldots, r \ , \\ q_c(i,i+1) & = & d_i + \vec{1}\{i = r-1\}u_{r-1} \ , & i = 0, 1, \ldots, r-1 \ , \\ q_c(i,r) & = & u_i \ , & i = 0, 1, \ldots, \min\{r-k, r-2\} \ , \\ q_c(i,i) & = & -(c_i + d_i + u_i) \ , & i = 0, 1, \ldots, r \ , \end{array} \tag{1}$$

where $c_i := (s+i)\alpha_1$, $d_i := (r-i)p\alpha_2$ and $u_i := \beta_c(r-i)\vec{1}\{i \leq r-k\}$ for $i \in \mathcal{T}$. Note that $\vec{Q}_c$ is not an infinitesimal generator since entries in its first row do not sum up to 0. From the theory of absorbing Markov chains we know that (e.g. [11, Lemma 2.2])

$$P(T_c(i) < x) = 1 - \vec{e}_i \cdot \exp(x\vec{Q}_c) \cdot \vec{1} \ , \quad x > 0 \ , \ i \in \mathcal{T} \ , \tag{2}$$

where $\vec{e}_i$ and $\vec{1}$ are vectors of dimension $r+1$; all entries of $\vec{e}_i$ are null except the i-th entry that is equal to 1, and all entries of $\vec{1}$ are equal to 1. In particular [11, p. 46]

$$\mathrm{E}[T_c(i)] = -\vec{e}_i \cdot \vec{Q}_c^{-1} \cdot \vec{1} \ , \quad i \in \mathcal{T} \ , \tag{3}$$

where the existence of $\vec{Q}_c^{-1}$ is a consequence of the fact that all states in $\mathcal{T}$ are transient [11, p. 45]. Let $T_c(i,j) = \int_0^{T_c(i)} \vec{1}\{X_c(t) = j\}dt$ be the total time spent by the CTMC in transient state j given that $X_c(0) = i$. It can also be shown that [7]

$$\mathrm{E}[T_c(i,j)] = -\vec{e}_i \cdot \vec{Q}_c^{-1} \cdot \vec{e}_j \ , \quad i,j \in \mathcal{T} \ . \tag{4}$$

Even when $\beta_c(0) = \cdots = \beta_c(r)$ an explicit calculation of either $P(T_c(i) < x)$, $\mathrm{E}[T_c(i)]$ or $\mathrm{E}[T_c(i,j)]$ is intractable, for any k in $\{1, 2, \ldots, r\}$. Numerical results for $\mathrm{E}[T_c(r)]$ and $P(T_c(r) > 10 \text{ years})$ are reported in Sect. 6.0.1 when $\beta_c(0) = \cdots = \beta_c(r)$.

4.2 Data Availability

In this section we introduce different metrics to quantify the availability of the block of data. The fraction of time spent by the absorbing Markov chain $\{X_c(t),\, t \geq 0\}$ in state j with $X_c(0) = i$ is $\mathrm{E}[(1/T_c(i)) \int_0^{T_c(i)} \vec{1}\{X_c(t) = j\}dt]$. However, since it is difficult to find a closed-form expression for this quantity, we will instead approximate it by the ratio $\mathrm{E}[T_c(i,j)]/\mathrm{E}[T_c(i)]$. With this in mind, we introduce

$$M_{c,1}(i) := \sum_{j=0}^{r} j \frac{\mathrm{E}[T_c(i,j)]}{\mathrm{E}[T_c(i)]} \ , \quad M_{c,2}(i) := \sum_{j=m}^{r} \frac{\mathrm{E}[T_c(i,j)]}{\mathrm{E}[T_c(i)]} \ , \quad i \in \mathcal{T} \ . \tag{5}$$

The first availability metric can be interpreted as the expected number of available redundant fragments during the block lifetime, given that $X_c(0) = i \in \mathcal{T}$. The second metric can be interpreted as the fraction of time when there are at least m redundant fragments during the block lifetime, given that $X_c(0) = i \in \mathcal{T}$. Both quantities $M_{c,1}(i)$ and $M_{c,2}(i)$ can be (numerically) computed from (3) and (4). Numerical results are reported in Sect. 6.0.2 for $i = r$ and $m = r - k$ in (5).

Since it is difficult to come up with an explicit expression for either metric $M_{c,1}(i)$ or $M_{c,2}(i)$, we make the assumption that parameters k and r have been selected so that the time before absorption is "large". This can be formalized, for instance, by requesting that $P(T_c(r) > q) > 1 - \epsilon$, where parameters q and ϵ are set according to the particular storage application(s). Instances are given in Sect. 6.0.2. In this setting, one may ignore the absorbing state a and represent the state of the storage system by a new irreducible and aperiodic – and therefore ergodic – Markov chain $\vec{\tilde{X}}_c := \{\tilde{X}_c(t),\, t \geq 0\}$ on the state-space $\mathcal{T}$. Let $\vec{\tilde{Q}}_c = [\tilde{q}_c(i,j)]_{0 \leq i,j \leq r}$ be its infinitesimal generator. Matrices $\vec{\tilde{Q}}_c$ and $\vec{Q}_c$, whose non-zero entries are given in (1), are identical except for $\tilde{q}_c(0,0) = -(u_0 + d_0)$. Until the end of this section we assume that $\beta_c(i) = \beta_c$ for $i \in \mathcal{T}$.

Let $\pi_c(i)$ be the stationary probability that $\vec{\tilde{X}}_c$ is in state i. Our objective is to compute $E[\tilde{X}_c] = \sum_{i=0}^{r} i\pi_c(i)$, the (stationary) expected number of available redundant fragments. To this end, let us introduce $f_c(z) = \sum_{i=0}^{r} z^i \pi_c(i)$, the generating function of the stationary probabilities $\vec{\pi}_c = (\pi_c(0), \pi_c(1), \ldots, \pi_c(r))$. Starting from the Kolmogorov balance equations $\vec{\pi}_c \cdot \vec{\tilde{Q}}_c = 0$, $\vec{\pi}_c \cdot \vec{1} = 1$, standard algebra yields

$$(\alpha_1 + p\alpha_2\, z)\tfrac{df_c(z)}{dz} = rp\alpha_2 f_c(z) - s\alpha_1 \tfrac{f_c(z) - \pi_c(0)}{z} + \beta_c \tfrac{f_c(z) - z^r}{1-z} - \beta_c \sum_{i=r-k+1}^{r} \tfrac{z^i - z^r}{1-z} \pi_c(i).$$

Letting $z = 1$ and using the identities $f_c(1) = 1$ and $df_c(z)/dz|_{z=1} = E[\tilde{X}_c]$, we find

$$\mathrm{E}[\tilde{X}_c] = \frac{r(p\alpha_2 + \beta_c) - s\alpha_1(1 - \pi_c(0)) - \beta_c \sum_{i=0}^{k-1} i\pi_c(r-i)}{\alpha_1 + p\alpha_2 + \beta_c} \ . \tag{6}$$

Unfortunately, it is not possible to find an explicit expression for $\mathrm{E}[\tilde{X}_c]$ since this quantity depends on the probabilities $\pi_c(0), \pi_c(r-(k-1)), \pi_c(r-(k-2)), \ldots, \pi_c(r)$, which cannot be computed in explicit form. If $k = 1$ then

$$\mathrm{E}[\tilde{X}_c] = \frac{r(p\alpha_2 + \beta_c) - s\alpha_1(1 - \pi_c(0))}{\alpha_1 + p\alpha_2 + \beta_c} \ , \tag{7}$$

which still depends on the unknown probability $\pi_c(0)$.

Below, we use a mean field approximation to develop an approximation formula for $\mathrm{E}[\tilde{X}_c]$ for $k = 1$, in the case where the maximum number of redundant fragments r is large. Until the end of this section we assume that $k = 1$. Using [9, Thm. 3.1] we know that, when r is large, the expected number of available redundant fragments at time t, $\mathrm{E}[\tilde{X}_c(t)]$, is solution of the following first-order differential (ODE) equation

$$\dot{y}(t) = -(\alpha_1 + p\alpha_2 + \beta_c)y(t) - s\alpha_1 + r(p\alpha_2 + \beta_c) \ .$$

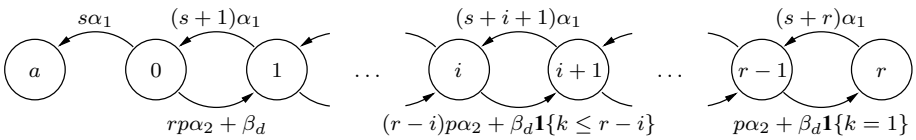


Figure 2: Transition rates of the absorbing Markov chain $\{X_d(t), t \geq 0\}$.

The equilibrium point of the above ODE is reached when time goes to infinity, which suggests to approximate $\mathrm{E}[\tilde{X}_c]$, when r is large, by

$$\mathrm{E}[\tilde{X}_c] \approx y(\infty) = \frac{r(p\alpha_2 + \beta_c) - s\alpha_1}{\alpha_1 + p\alpha_2 + \beta_c} \quad . \tag{8}$$

Observe that this simply amounts to neglect of the probability $\pi_c(0)$ in (7) for large r.

5 Distributed Repair Systems

In this section, we address the performance analysis of the distributed implementation of the P2P storage system, as described in Sect. 3. Recall that in the distributed setting, as soon as k fragments become unreachable, secure agents running on k new peers simultaneously initiate the recovery of one fragment each.

5.1 Data Lifetime

Since the analysis is very similar to the analysis in Sect. 4 we will only sketch it. Alike in the centralized implementation, the state of the system can be represented by an absorbing Markov chain $\vec{X}_d := \{X_d(t),\ t \geq 0\}$, taking values in the set $\{a\} \cup \mathcal{T}$ (recall that $\mathcal{T} = \{0, 1, \dots, r\}$). State a is the absorbing state indicating that the block of data is lost (less than s fragments available), and state $i \in \mathcal{T}$ gives the number of available redundant fragments. The non-zero transition rates of this absorbing Markov chain are displayed in Fig. 2. Non-zero entries of the matrix $\vec{Q}_d = [q_d(i,j)]_{0\leq i,j\leq r}$ associated with the absorbing Markov chain $\vec{X}_d$ are given by

$$\begin{array}{lcll} q_d(i,i-1) & = & c_i \ , & i = 1,2,\dots,r \ , \\ q_d(i,i+1) & = & d_i + w_i \ , & i = 0,1,\dots,r-1 \ , \\ q_d(i,i) & = & -(c_i + d_i + w_i) \ , & i = 0,1,\dots,r \ , \end{array}$$

with $w_i := \beta_d \vec{1}\{i \leq r-k\}$ for $i = 0, 1, \dots, r$, where c_i and d_i are defined in Sect. 4. Note that letting $s = 1$, $p = 0$ and $k = 1$ yields the model of [14]. Introduce $T_d(i) := \inf\{t \geq 0 \ : \ X_d(t) = a\}$ the time until absorption in state a given that $X_d(0) = i$, and let $T_d(i,j)$ be the total time spent in transient state j given that $X_d(0) = i$. The distribution $P(T_d(i) < x)$, $\mathrm{E}[T_d(i)]$ and $\mathrm{E}[T_d(i,j)]$ are given by (2), (3) and (4), respectively, after replacing $\vec{Q}_c$ by $\vec{Q}_d$. Alike for $\vec{Q}_c$ it is not tractable to explicitly invert $\vec{Q}_d$. Numerical results for $\mathrm{E}[T_d(r)]$ and $P(T_d(r) > 1 \text{ year})$ are reported in Sect. 6.0.1.

5.2 Data Availability

As motivated in Sect. 4.2 the metrics

$$M_{d,1}(i) := \sum_{j=0}^{r} j \frac{\mathrm{E}[T_d(i,j)]}{\mathrm{E}[T_d(i)]} \quad , \quad M_{d,2}(i) := \sum_{j=m}^{r} \frac{\mathrm{E}[T_d(i,j)]}{\mathrm{E}[T_d(i)]} \quad , \tag{9}$$

can be used to quantify the data availability in distributed-recovery P2P storage systems. Numerical results are given in Sect. 6.0.2. Similar to what was done in Sect. 4.2, let us assume that parameters r and k have been tuned so that the time before absorption is "long". If so, then as an approximation one can consider that absorbing state a can no longer be reached. The Markov chain $\vec{X}_d$ becomes an irreducible, aperiodic Markov chain on the set $\mathcal{T}$, denoted $\vec{\tilde{X}}_d$. More precisely, it becomes a birth and death process (see Fig. 2). Let $\pi_d(i)$ be the stationary probability that $\vec{\tilde{X}}_d$ is in state i, then (e.g. [8])

$$\pi_d(i) = \left[1 + \sum_{i=1}^{r} \prod_{j=0}^{i-1} \frac{d_j + w_j}{c_{j+1}} \right]^{-1} \cdot \prod_{j=0}^{i-1} \frac{d_j + w_j}{c_{j+1}} \quad , \quad i \in \mathcal{T} \ . \tag{10}$$

From (10) we can derive the expected number of available redundant fragments through the formula $\mathrm{E}[\tilde{X}_d] = \sum_{i=0}^{r} i\pi_d(i)$. Numerical results for $\mathrm{E}[\tilde{X}_d]$, or more precisely, for its deviation from $M_{d,1}(r)$ are reported in Sect. 6.0.2.

6 Numerical Results

In this section we provide numerical results using the Markovian analysis presented earlier. Our objectives are to characterize the performance metrics defined in the paper against the system parameters and to illustrate how our models can be used to engineer the storage systems.

Throughout the numerical computations, we consider storage systems for which the dynamics have either one or two timescales, and whose recovery implementation is either centralized or distributed. Dynamics with two timescales arise in a *company context* in which disconnections are chiefly caused by failures or maintenance conditions. This yields slow peer dynamics and significant data losses at disconnected peers. However, the recovery process is particularly fast. Storage systems deployed over a wide area network, hereafter referred to as the *Internet context*, suffer from both fast peer dynamics and a slow recovery process. However, it is highly likely that peers will still have the stored data at reconnection.

The initial number of fragments is set to $s = 8$, deriving from the fact that fragment and block sizes in P2P systems are often set to $64KB$ and $512KB$ respectively (block sizes of $256KB$ and $1MB$ are also found). The recovery rate in the centralized scheme is made constant. The amount of redundancy r will be varied from 1 to 30 and for each value of r, we vary the threshold k from 1 to r. In the company context we set $1/\alpha_1 = 5$ days, $1/\alpha_2 = 2$ days, $p = 0.4$, $1/\beta_c = 11$ minutes and $1/\beta_d = 10$ minutes. In the Internet context we set $1/\alpha_1 = 5$ hours, $1/\alpha_2 = 3$ hours, $p = 0.8$, $1/\beta_c = 34$ minutes and $1/\beta_d = 30$ minutes. This setting of the parameters is mainly for illustrative purposes. Recall that the recovery process accounts for the time needed to store the reconstructed data on the local (resp. remote) disk in the distributed (resp. centralized) scheme. Because of the network latency, we will always have $\beta_c < \beta_d$.

6.0.1 The Conditional Block Lifetime

We have computed the expectation and the *complementary* cumulative distribution function (CCDF) of $T_c(r)$ and $T_d(r)$ using (3) and (2) respectively. The results are reported in Figs. 3 and 4 respectively. The discussion on Fig. 4 comes later on.

We see from Fig. 3 that $\mathrm{E}[T_c(r)]$ and $\mathrm{E}[T_d(r)]$ increase roughly exponentially with r and are decreasing functions of k. When the system dynamics has two timescales like in a company context, the expected lifetime decreases exponentially with k whichever the recovery mechanism considered. Observe in this case how large the block lifetime can become for certain values of r and k. Observe also that the centralized scheme achieves higher block lifetime than the distributed scheme unless $k = 1$ and $r = 1$ (resp. $r \leq 6$) in the Internet (resp. company) context.

6.0.2 The Availability Metrics

We have computed the availability metrics $M_{c,1}(r)$, $M_{d,1}(r)$ and $M_{c,2}(r)$ and $M_{d,2}(r)$ with $m = r - k$ using (5) and (9). The results are reported in Figs. 5 and 6 respectively. The discussion on Fig. 6 comes later on.

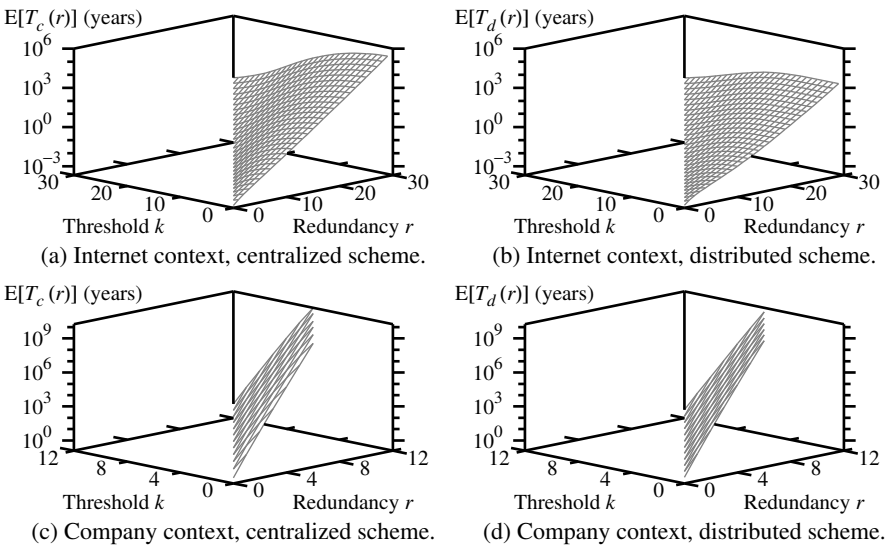
We see from Fig. 5 that alike for the lifetime, metrics $M_{c,1}(r)$ and $M_{d,1}(r)$ increase exponentially with r and decrease as k increases. The shape of the decrease depends on which recovery scheme is used within which context. We again find that the centralized scheme achieves higher availability than the distributed scheme unless $k = 1$ and $r = 1$ (resp. $r \leq 26$) in the Internet (resp. company) context.

Regarding $M_{c,2}(r)$ and $M_{d,2}(r)$, we have found them to be larger than 0.997 for any of the considered values of r and k in the company context. This result is expected because of the two timescales present in the system. Recall that in this case the recovery process is two-order of magnitude faster than the peer dynamics. The results corresponding to the Internet context can be seen in Fig. 6.

Last, we have computed the expected number of available redundant fragments $\mathrm{E}[\tilde{X}_c]$ and $\mathrm{E}[\tilde{X}_d]$. The results are almost identical to the ones seen in Fig. 5. The deviation between $\mathrm{E}[\tilde{X}_d]$ and $M_{d,1}(r)$ in the Internet context is the largest among the four cases. Figure 7(a) delimits the regions where the deviation is within certain value ranges. For instance, in region V the deviation is smaller than 1‰. If the storage system is operating with values of r and k from this region, then it will be attractive to evaluate the data availability using $\mathrm{E}[\tilde{X}_d]$ instead of $M_{d,1}(r)$.

6.0.3 Engineering the system

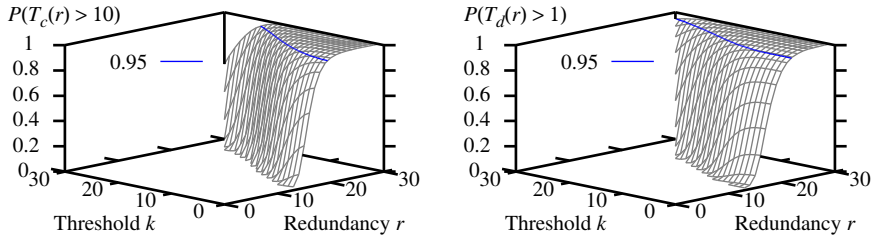
Using our theoretical framework it is easy to tune the system parameters for fulfilling predefined requirements. As an illustration, Fig. 7(b) displays three contour lines of the CCDF of the lifetime $T_c(r)$ at point $q = 10$ years (see Fig. 4(a)) and two contour



(a) Internet context, centralized scheme. (b) Internet context, distributed scheme.

(c) Company context, centralized scheme. (d) Company context, distributed scheme.

Figure 3: Expected lifetime $\mathrm{E}[T_c(r)]$ and $\mathrm{E}[T_d(r)]$ (expressed in years) versus r and k.



(a) Internet context, centralized scheme. (b) Internet context, distributed scheme.

Figure 4: (a) $P(T_c(r) > 10$ years) and (b) $P(T_d(r) > 1$ year) versus r and k.

lines of the availability metric $M_{c,2}(r)$ with $m = r - k$ (see Fig. 6(a)). Consider point A which corresponds to $r = 27$ and $k = 7$. Selecting this point as the operating point of the storage system will ensure that $P(T_c(r) > 10) = 0.999$ and $M_{c,2}(r) = 0.8$. In other words, when $r = 27$ and $k = 7$, only 1‰ of the stored blocks would be lost after 10 years and for 80% of a block lifetime there will be 20 ($= r - k$) or more redundant fragments from the block available in the system.

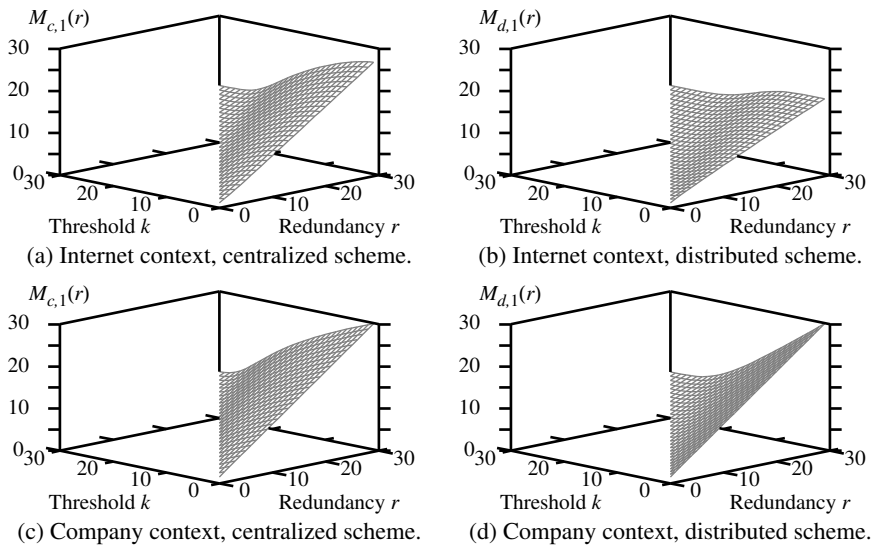
One may be interested in only guaranteeing large data lifetime. Values of r and k are then set according to the desired contour line of the CCDF of data lifetime. Smaller threshold values enable smaller amounts of redundant data at the cost of higher bandwidth utilization. The trade-off here is between efficient storage use (small r) and efficient bandwidth use (large k).

7 Conclusion

We have proposed simple Markovian analytical models for evaluating the performance of two approaches for recovering lost data in distributed storage systems. One approach relies on a centralized server to recover the data; in the other approach new peers perform this task in a distributed way. We have analyzed the lifetime and the availability of data achieved by both centralized- and distributed-repair systems through Markovian analysis and fluid approximations. Numerical computations have been undertaken to support the performance analysis. Using our theoretical framework it is easy to tune the system parameters for fulfilling predefined requirements. Concerning future work, current efforts focus on modeling storage systems where peer lifetimes are either Weibull or hyperexponentially distributed (see [12]).

References

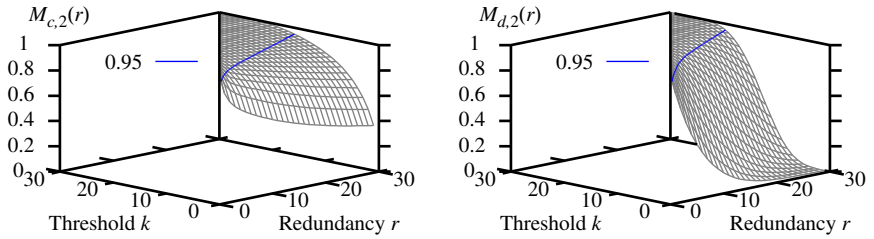
[1] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G.M. Voelker. Total Recall: System support for automated availability management. In *Proc. of ACM/USENIX NSDI '04, San Francisco, California*, pages 337–350, March 2004.



(a) Internet context, centralized scheme. (b) Internet context, distributed scheme.

(c) Company context, centralized scheme. (d) Company context, distributed scheme.

Figure 5: Availability metrics $M_{c,1}(r)$ and $M_{d,1}(r)$ versus r and k.



(a) Internet context, centralized scheme. (b) Internet context, distributed scheme.

Figure 6: Availability metrics (a) $M_{c,2}(r)$ and (b) $M_{d,2}(r)$ versus r and k with $m = r - k$.

[2] C. Blake and R. Rodrigues. High availability, scalable storage, dynamic peer networks: Pick two. In *Proc. of HotOS-IX, Lihue, Hawaii*, May 2003.

[3] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *LNCS 2009, Proc. of Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66. Springer, July 2000.

[4] F. Dabek, M.F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proc. of ACM SOSP '01, Banff, Canada*, pages 202–215, October 2001.

[5] Farsite: Federated, available, and reliable storage for an incompletely trusted environment. `http://research.microsoft.com/Farsite/`, 2006.

[6] A.V. Goldberg and P.N. Yianilos. Towards an archival Intermemory. In *Proc. of ADL '98, Santa Barbara, California*, pages 147–156, April 1998.

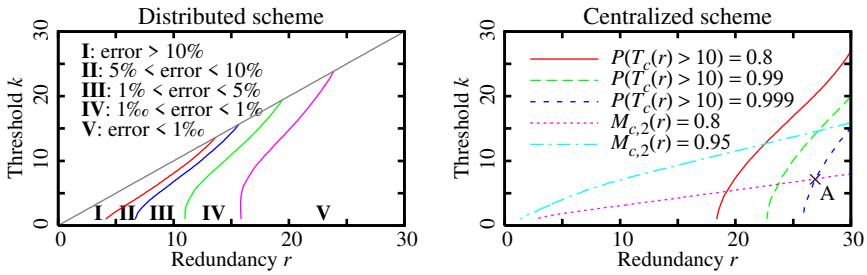
[7] C. Grinstead and J. Laurie Snell. *Introduction to Probability*. American Math. Soc., 1997.

[8] L. Kleinrock. *Queueing Systems, Vol. 1*. J. Wiley, New York, 1975.

[9] T.G. Kurtz. Solutions of ordinary differential equations as limits of pure jump markov processes. *Journal of Applied Probability*, 7(1):49–58, April 1970.

[10] W.K. Lin, D.M. Chiu, and Y.B. Lee. Erasure code replication revisited. In *Proc. of IEEE P2P '04, Zurich, Switzerland*, pages 90–97, August 2004.

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



(a) Relative error $|M_{d,1} - \mathrm{E}[\tilde{X}_d]|/M_{d,1}$.

(b) Selection of r and k according to predefined requirements.

Figure 7: Numerical results for the Internet context.

[12] D. Nurmi, J. Brevik, and R. Wolski. Modeling machine availability in enterprise and wide-area distributed computing environments. Technical Report CS2003-28, University of California Santa Barbara, 2003.

[13] The OceanStore project: Providing global-scale persistent data. `http://oceanstore.cs.berkeley.edu/`, 2005.

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

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

[16] 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, Banff, Canada*, pages 188–201, October 2001.

[17] G. Utard and A. Vernois. Data durability in peer to peer storage systems. In *Proc. of IEEE GP2PC '04, Chicago, Illinois*, April 2004.

[18] H. Weatherspoon and J. Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In *Proc. of IPTPS '02, Cambridge, Massachusetts*, March 2002.