



# Lifetime and availability of data stored on a P2P system: Evaluation of recovery schemes

Abdulhalim Dandoush, Sara Alouf, Philippe Nain

## ► To cite this version:

Abdulhalim Dandoush, Sara Alouf, Philippe Nain. Lifetime and availability of data stored on a P2P system: Evaluation of recovery schemes. [Research Report] RR-7170, INRIA. 2010, pp.37. inria-00448100v2

**HAL Id: inria-00448100**

**<https://inria.hal.science/inria-00448100v2>**

Submitted on 29 Apr 2010

**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.



INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

*Lifetime and availability of data stored on a P2P  
system: Evaluation of redundancy and recovery  
schemes*

Abdulhalim Dandoush — Sara Alouf — Philippe Nain

N° 7170

April 2010

Thème COM

*Rapport  
de recherche*





## Lifetime and availability of data stored on a P2P system: Evaluation of redundancy and recovery schemes

Abdulhalim Dandoush , Sara Alouf , Philippe Nain

Thème COM — Systèmes communicants  
Projet MAESTRO

Rapport de recherche n° 7170 — April 2010 — 34 pages

**Abstract:** This report studies the performance of Peer-to-Peer storage and backup systems (P2PSS). These systems are based on three pillars: data fragmentation and dissemination among the peers, redundancy mechanisms to cope with peers churn, and repair mechanisms to recover lost or temporarily unavailable data. Usually, redundancy is achieved either by using replication or by using erasure codes. A new class of network coding (regenerating codes) has been proposed recently. Therefore, we will adapt our work to these three redundancy schemes. We introduce two mechanisms for recovering lost data and evaluate their performance by modeling them through absorbing Markov chains. Specifically, we evaluate the quality of service provided to users in terms of durability and availability of stored data for each recovery mechanism and deduce the impact of its parameters on the system performance. The first mechanism is centralized and based on the use of a single server that can recover multiple losses at once. The second mechanism is distributed: reconstruction of lost fragments is iterated sequentially on many peers until the required level of redundancy is attained. The key assumptions made in this work, in particular, the assumptions made on the recovery process and peer on-times distribution, are in agreement with the analysis in [11] and in [20] respectively. The models are thereby general enough to be applicable to many distributed environments as shown through numerical computations. We find that, in stable environments such as local area or research institute networks where machines are usually highly available, the distributed-repair scheme in erasure-coded systems offers a reliable, scalable and cheap storage/backup solution. For the case of highly dynamic environments, in general, the distributed-repair scheme is inefficient, in particular to maintain high data availability, unless the data redundancy is high. Using regenerating codes overcomes this limitation of the distributed-repair scheme. P2PSS with centralized-repair scheme are efficient in any environment but have the disadvantage of relying on a centralized authority. However, the analysis of the overhead cost (e.g. computation, bandwidth cost and complexity) resulting from the different redundancy schemes with respect to their advantages (e.g. simplicity), is left for future work.

**Key-words:** Peer-to-Peer network, distributed storage system, performance evaluation, absorbing Markov chain, data availability, data durability, recovery process

## Disponibilité et longévité des données stockées dans les systèmes de pairs : Evaluation de mécanismes de redondance et de recouvrement

**Résumé :** Ce rapport étudie les performances de systèmes pair-à-pair de stockage des données (P2PSS). Ces systèmes reposent sur trois piliers: la fragmentation des données et leur dissémination chez les pairs, la redondance des données afin de faire face aux éventuelles indisponibilités des pairs et l'existence d'un mécanisme de réparation ou de recouvrement des données perdues ou temporairement indisponibles. Nous introduisons deux mécanismes de recouvrement des données perdues et en évaluons les performances en les modélisant par des chaînes de Markov absorbantes. Plus précisément, nous évaluons la qualité du service rendu aux utilisateurs en terme de longévité et de disponibilité des données de chaque mécanisme et en déduisons l'impact de ses paramètres sur ses performances. Le premier mécanisme est centralisé et repose sur l'utilisation d'un unique serveur pour la reconstruction des fragments de donnée perdus. Le second mécanisme est distribué : la reconstruction des fragments perdus met en oeuvre, séquentiellement, plusieurs pairs et s'arrête dès que le niveau de redondance requis est atteint. Les principales hypothèses faites dans ce travail, notamment celles portant sur la distribution du processus de recouvrement et de disponibilité des pairs, sont en accord avec les hypothèses dans [11, 20]. Nos modèles, qui généralisent ceux dans [10], sont suffisamment génériques pour s'appliquer à différents environnements distribués, comme le montrent nos applications numériques. Nous constatons que dans des environnements stables comme les réseaux locaux des instituts de recherche, où les machines sont généralement très disponibles, le mécanisme de recouvrement distribué offre une solution de stockage fiable, évolutive et peu coûteuse. Pour des réseaux très dynamiques l'efficacité du mécanisme distribué est inversement proportionnelle au niveau de redondance des données, tout particulièrement pour ce qui concerne leur disponibilité. Les systèmes de stockage qui implémentent le mécanisme de recouvrement centralisé sont efficaces dans n'importe quel environnement, leur talon d'achille étant les pannes du serveur qui rendent inopérantes la reconstruction des données perdues.

**Mots-clés :** systèmes pair-à-pair, évaluation de performance, chaîne de Markov absorbante, approximation champ moyen

## 1 Introduction

Conventional storage solutions rely on robust dedicated servers and magnetic tapes on which data are stored. These equipments are reliable, but they are also expensive and do not scale well. The growth of storage volume, bandwidth, and computational resources for PCs has fundamentally changed the way applications are constructed. Almost 10 years ago, a new network paradigm has been proposed where computers can build a virtual network (called overlay) on top of another network or an existing architecture (e.g. Internet). This new network paradigm has been labeled peer-to-peer (P2P) distributed network. A peer in this paradigm is a computer that play the role of both supplier and consumer of resources, in contrast to the traditional client-server model where only servers supply, and computers consume. Applications that use this distributed network provides enhanced scalability and service robustness as all the connected computers or peers provide some services. Peers in the overlay can be thought of as being connected by virtual or logical links, each of which corresponds to a path, perhaps through many physical links, in the underlying network. As already mentioned, each peer receives/provides a service from/to other peers through the overlay network; examples of such services are computing (sharing the capacity of its central processing unit), data upload (sharing its bandwidth capacity), data storage (sharing its free storage space), as well as support to locate resources, services and other peers.

This P2P model has proved to be an alternative to the Client/Server model and a promising paradigm for Grid computing, file sharing, voice over IP, backup and storage applications. Some of the recent efforts for building highly available storage or backup system based on the P2P paradigm include Intermemory [13], OceanStore [18], CFS [9], PAST [24], Total Recall [5], UbiStorage [27] and Tahoe [28]. Although scalable and economically attractive compared to traditional storage/backup systems, these P2P systems pose many problems such as reliability, data availability and confidentiality.

We can distinguish between backup and storage systems. P2P backup systems aim to provide long data lifetime without constraints on the data availability level or the reconstruction time. For this reason, the backup system designers are interested in the permanent departures of peers rather than the intermediate disconnections, on the contrary to storage systems, even if the disconnections durations were long.

### 1.1 Redundancy schemes

In a P2P network, peers are free to leave and join the system at any time. As a result of the intermittent availability of peers, ensuring high availability of the stored data is an interesting and challenging problem. To ensure data reliability and availability in such dynamic systems, redundant data is inserted into the system. Existing systems achieve redundancy either by replication, where there are two replication levels, or by erasure codes (e.g. [23, 6]). The two replication levels are as follows:

- *The whole-file-level replication scheme.* A file  $f$  is replicated  $r$  times over  $r$  different peers (as in PAST [24]) so that the tolerance against failures or peers departure is equal

to  $r$ . The ratio  $1/\{r+1\}$  defines the useful storage space in the system. Hereafter, we will refer to this replication scheme as *replication*.

- *The fragment-level replication scheme.* This scheme consists of dividing the file  $f$  into  $s$  equally sized fragments, and then make  $r$  copies of each of them and place one fragment copy per peer, as in CFS [9].

The erasure code (EC) scheme consists of dividing the file  $f$  into  $b$  equally sized blocks (say  $S_B$  bits). Each block of data  $D$  is partitioned into  $s$  equally sized fragments (say  $F_{EC} = S_B/s$  bits) to which, using one of the erasure codes scheme (e.g. [23, 6]),  $r$  redundant fragments are added of the same size. To download a block of data, any  $s$  fragments are needed out of the  $s+r$  (downloading the size of the original block  $S_B$ ). Recovering any fragment (if it is lost) or adding a new redundant fragment of a given block of data requires the download of any other  $s$  fragments out of the available fragments of that block (downloading again the size of the original block  $S_B$ ). Therefore, for each stored block of data, the tolerance against failures or peers departure is equal to  $r$ . The useful storage space in the system is defined by the ratio  $s/(s+r)$ . Intermemory [13, 7], OceanStore [18], Total Recall [5] and UbiStorage [27] are some examples of existing P2P systems that use erasure coding mechanisms to provide some level of system reliability and data availability.

For the same amount of redundancy, erasure codes provide higher availability of data than replication [29]. In [3], the authors show that an erasure codes scheme makes backup systems more scalable than replication and block-level replication schemes as the required availability gets higher. They show as well that the scalability of the block-level scheme with respect to the total storage required is even lower than that of the replication scheme.

A new class of codes, so-called regenerating codes (RC) has been proposed recently in [12]. RC can be considered a generalization of erasure code (EC), which reduces the communication cost of EC by slightly increasing the storage cost. The size of fragments in RC is larger of that in EC. In [12], the authors consider in *Theorem 1, p. 5* a simple scheme in which they require that any  $s$  fragments (the minimum possible) can reconstruct the original block of data. All fragments have equal size  $F_{RC} = \alpha * S_B$ , where  $S_B$  stands for the size of the given block of data to be stored. A newcomer (a new peer in our notation) produces a new redundant fragment by connecting to any  $s$  nodes and downloading  $\alpha S_B/s$  bits from each. In this theorem, the authors assume that the source node of the block of data will store initially  $n$  fragments of size  $\alpha S_B$  bits on  $n$  storage nodes. In addition, newcomers arrive sequentially and each one connects to an arbitrary  $k$ -subset of previous nodes (including previous newcomers). They define  $\alpha_c := \frac{s}{s^2 - s + 1}$  to be, in the worth case, the lower bound on the minimal amount of data that a newcomer must download. The worth case is occurred when a data collector (client) need to recover the original block of data from only newcomers. In general, if  $\alpha \geq \alpha_c$  there exists a linear network code so that all data collectors can reconstruct the considered block by downloading  $s$  fragments from any  $s$  nodes. So, using this simple scheme of RC, adding a new redundant fragment of a given block requires a new peer to download  $1/s$  percent of  $s$  stored fragments ( $\alpha S_B/s$  of each) so that the new peer regenerates one random linear combination of the parts of fragments



already downloaded; the new peer will store all the downloaded data whose size is equal to the size of the stored fragments instead to download the equivalent of original block size, in the case of EC, to regenerate one fragment and deleting later the downloaded fragments. In the same way, downloading the block, by a data collector, in EC requires the download of its size ( $s * F_{EC} = S_B$ ), where in RC, it requires the download of  $s * F_{RC} = s * \alpha * S_B = \beta S_B$ , where  $\beta > 1$ . The authors show that  $\beta \rightarrow 1$  as  $s \rightarrow \infty$ . Until the time of writing this report, the regenerating codes is not yet used in any P2P system. However, we will show through numerical results that RC is a promising redundancy scheme for P2P storage and backup systems.

## 1.2 Recovery mechanisms and policies

Using redundancy mechanisms without repairing lost data is not efficient, as the level of redundancy decreases when peers leave the system. Consequently, P2P storage/backup 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*. A centralized server can recover at once multiple losses of the same document in the centralized-recovery scheme. In the distributed case, each new host—thanks to its secure agent—recovers only one loss per document.

Regardless of the recovery mechanism used, two repair policies can be enforced. In the *eager* policy, when the system detects that one host has left the network, it immediately initiates the reconstruction of the lost data, and stores it on a new peer upon recovery. This policy is simple but makes no distinction between permanent departures that need to be recovered, and transient disconnections that do not.

Having in mind that connections may experience temporary, as opposed to permanent failures, one may want to deploy a system that defers the repair beyond the detection of a first loss of data. This alternative policy, called *lazy*, inherently uses less bandwidth than the eager policy. However, it is obvious that an extra amount of redundancy is necessary to mask and to tolerate host departures for extended periods of time.

## 1.3 Contributions

The aim of this report is to develop mathematical models to evaluate fundamental performance metrics (data lifetime and availability) of P2PSS. Our contributions are as follows:

- Analysis of centralized and distributed recovery mechanisms.
- Proposition of a general model that captures the behavior of eager/lazy repair policies and the three replication schemes, and accommodates both temporary and permanent disconnections of peers.

- Numerical investigation using realistic parameters values to support the mathematical models and to compare erasure codes with regenerating codes.
- Provision of guidelines on how to engineer a P2P backup or storage system in order to satisfy given requirements.

In the following, Section 2 reviews related work and Section 3 introduces the assumptions and notation used throughout the report. In Section 4, we discuss some preliminary material to be used in subsequent sections. Sections 5 and 6 are dedicated to the modeling of the centralized- and distributed-recovery mechanism, respectively. In Section 8, we provide some numerical results showing the performance of the centralized and decentralized schemes when either erasure codes or regenerating codes are used. Section 9 concludes the report.

## 2 Related Work and Background

Although the literature on the architecture and file system of distributed backup and storage systems is abundant, most of these systems are configured statically to provide durability and/or availability with only a cursory understanding of how the configuration will impact overall performance. Some systems allow data to be replicated and cached without constraints on the storage overhead or on the frequency at which data are cached or recovered. These yield to waste of bandwidth and storage volume and do not provide a clear predefined durability and availability level. Hence, the importance of the thorough evaluation of P2P storage systems before their deployment.

There have been recent modeling efforts focusing on the performance analysis of P2P backup and storage systems in terms of data durability and availability such as [22, 1] and [?]. However, in all these models, findings and conclusions rely on the assumption that the peers availability and the recovery process are exponentially distributed with some parameter for the sake of simplification or for the lack of works characterizing their distribution under realistic settings and assumptions.

Characterizing peers availability both in local and wide area environments has been the focus of [20]. In this paper, Nurmi, Brevik and Wolski investigate three sets of data, each measuring machine availability in a different setting, and perform goodness-of-fit tests on each data set to assess which out of four distributions best fits the data. They have found that a hyper-exponential model fits more accurately the machine availability durations than the exponential, Pareto, or Weibull distribution.

To understand how the recovery process could be better modeled, we performed a packet-level simulation analysis of the download and the recovery processes in erasure-coded systems; cf. [11]. We ran several experiments and collected a large number of samples of these processes in a large variety of scenarios. We used expectation maximization and least square estimation algorithms to fit the empirical distributions and tested the goodness of our fits using statistical (Kolmogorov-Smirnov test) and graphical methods. We found that the download time of a fragment of data located on a single peer follows approximately an

exponential distribution. We also found that the recovery time essentially follows a hypo-exponential distribution with many distinct phases. We found that the download time of a fragment of data located on a single peer follows approximately an exponential distribution. We also found that the recovery time essentially follows a hypo-exponential distribution with many distinct phases.

In light of the conclusions of [20], namely, that machine availability is modeled with a hyper-exponential distribution, and building on the findings of [11] we will propose in this report general and accurate models that are valid under different distributed environments.

### 3 System Description and Assumptions

In the following, we will distinguish the *peers*, which are computers where data is stored and which form a storage system, from the *users* whose objective is to retrieve the data stored in the storage system.

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

- A block of data  $D$  is partitioned into  $s$  equally sized fragments to which, using erasure codes or regenerating code,  $r$  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 a *persistence* probability  $p$  (and with probability  $1 - p$  they are 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 the two different repair policies: the *eager* and the *lazy* repair policies. Recall that in the lazy policy, the repair is delayed until the number of unavailable fragments reaches a given threshold, denoted  $k$ . 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  $k \in \{1, 2, \dots, r\}$ , where  $k$  can take any value in the set  $\{2, \dots, r\}$  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 the reconstructed fragments *in parallel* onto as many new peers for storage. The central authority updates the database recording fragments locations as soon as all uploads terminate. 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) ends executing when the download (resp. upload) of the last fragment is completed.

**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 (or  $s$  parts of  $s$  fragments if RC is used) 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) subsequently discards the  $s$  downloaded fragments, in the case of erasure code, so as to meet the privacy constraint that only one fragment of a block of data may be 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. We will thus consider the recovery process to end when the download of the last fragment (out of  $s$ ) is completed.

**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: (off-times)** We assume that successive durations of off-times of a peer are independent and identically distributed (iid) random variables (rvs) with a common exponential distribution function with parameter  $\lambda > 0$ .

Assumption 1 is in agreement with the analysis in [22].

**Assumption 2: (on-times)** We assume that successive durations of on-times of a peer are iid rvs with a common hyper-exponential distribution function with  $n$  phases; the parameters of phase  $i$  are  $\{p_i, \mu_i\}$ , with  $p_i$  the probability that phase  $i$  is selected and  $1/\mu_i$  the mean duration of phase  $i$ . We naturally have  $\sum_{i=1}^n p_i = 1$ .

Assumption 2, with  $n > 1$ , is in agreement with the analysis in [20]; when  $n = 1$ , it is in agreement with the analysis in [22].

**Assumption 3: (independence)** Successive on-times and off-times are assumed to be independent. Peers are assumed to behave independently of each other.

**Assumption 4: (download/upload durations)** We assume that successive download (resp. upload) durations of a fragment are iid rvs with a common exponential distribution function with parameter  $\alpha$  (resp.  $\beta$ ). We further assume that concurrent fragments downloads/uploads are not correlated.

Assumption 4 is supported by our findings in [11]. As already mentioned, the fragment download/upload time was found to follow approximately an exponential distribution. As for the concurrent downloads/uploads, we have found in simulations that these are weakly correlated and close to be “independent” as long as the total workload is equally distributed over the active peers. There are two main reasons for the weak correlation between concurrent downloads/uploads as observed in simulations: (i) the good connectivity of nowadays core networks and (ii) the asymmetry in peers upstream and downstream bandwidths, as on average, a peer tends to have higher downstream than upstream bandwidth [25, 15]. So, as the bottleneck would be the upstream capacity of peers, the fragment download times are close to be iid rvs.

A consequence of Assumption 4 is that each of the block download time and the durations of the centralized and the distributed recovery processes is a rv following a hypo-exponential distribution [16]. Indeed, each of these durations is the summation of *independently* distributed exponential rvs ( $s$  for the block download and in the distributed scheme, and  $s + k$  in the centralized scheme if  $k$  fragments are to be reconstructed) having each its own rate. This is a fundamental difference with [1] where the recovery process is assumed to follow an exponential distribution.

It is worth mentioning that the simulation analysis of [11] has concluded that in most cases the recovery time follows roughly a hypo-exponential distribution. This result is

Table 1: System parameters.

$D$	Block of data
$s$	Original number of fragments for each block of data
$r$	Number of redundant fragments
$k$	Threshold of the recovery process
$p$	Persistence probability
$\lambda$	Rate at which peers rejoin the system
$\{p_i, \mu_i\}_{i=1,\dots,n}$	Parameters of the peers failure process
$\alpha$	Download rate of a piece of data (fragment)
$\beta$	Upload rate of a fragment in the centralized-repair scheme

expected as long as fragments downloads/uploads are exponentially distributed and very weakly correlated. It was also found in [11] that a hypo-exponential model gives a more reasonable approximation of the recovery process than an exponential model even in cases when the null hypothesis is rejected.

Given Assumptions 1–4, the models developed in this report are more general and/or more realistic than those in [22, 1, 10]. Table 1 recapitulates the parameters introduced in this section. We will refer to  $s$ ,  $r$  and  $k$  as the *protocol* parameters,  $p$ ,  $\lambda$  and  $\{p_i, \mu_i\}_{i=1,\dots,n}$  as the *peers* parameters, and  $\alpha$  and  $\beta$  as the *network* parameters.

## 4 Preliminaries and Notation

We will focus in this section on the dynamic of peers in the storage system. In particular, we are interested in computing the stationary distribution of peers. According to Assumptions 1–3 in the previous section, each time a peer rejoins the system, it picks its on-time duration from an exponential distribution having parameter  $\mu_i$  with probability  $p_i$ , for  $i \in [1..n]$ . In other words, a peer can stay connected for a short time in a session and for a long time in another one.

This dynamicity can be modeled as a general queueing network with an arbitrary but finite number  $n$  of different classes of customers (peers) and an infinite number of servers. In this network, a new customer enters directly, with probability  $p_i$ , a server with a service rate  $\mu_i$ . Define  $PI(\vec{n} = (n_1, \dots, n_n)) := \lim_{t \rightarrow \infty} P(N_1(t) = n_1, \dots, N_n(t) = n_n)$  to be the joint distribution function of the number of customers of class  $1, \dots, n$  in steady-state (or, equivalently, the number of busy servers) where  $N_i(t)$  is the number of peers of class  $i$  in the system at time  $t$  for  $i = 1, \dots, n$ . We have the following known results [2, 17]:

$$PI(\vec{n}) = 1/G \prod_{i=1}^n \frac{\rho_i^{n_i}}{n_i!}$$

where  $\rho_i = \lambda p_i / \mu_i$  is the rate at which work enters class  $i$  and  $G$  is the normalizing constant.

$$G = \sum_{\vec{n} \in \mathbb{N}^n} \prod_{i=1}^n \frac{\rho_i^{n_i}}{n_i!}$$

Denote the expected number of customers of class  $i$  in the system by  $E[n_i]$  where

$$E[n_i] = \sum_{\vec{n} \in \mathbb{N}^n} n_i PI(\vec{n}) = \rho_i \prod_{l=1}^n e^{\rho_l}, \quad \text{for } i = 1, \dots, n$$

For later use, we will compute the probability of selecting a *new* peer in phase  $i$ , denoted by  $R(i)$ , or equivalently the percentage of the connected peers in phase  $i$  as follows:

$$R(i) = \frac{E[n_i]}{\sum_{l=1}^n E[n_l]} = \frac{\rho_i}{\sum_{l=1}^n \rho_l} = \frac{p_i/\mu_i}{\sum_{l=1}^n p_l/\mu_l} \quad (1)$$

We introduce as well functions  $S$  and  $f$  such that for a given  $n$ -tuple  $\vec{a} = (a_1, \dots, a_n)$ ,  $S(\vec{a}) := \sum_{i=1}^n a_i$  and  $f_i(\vec{a}) := a_i/S(\vec{a})$ .

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) recovery 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; and the notation  $\vec{0}$  refers to a null *row* vector of appropriate dimension.  $\mathbb{1}\{A\}$  is the characteristic function of event  $A$ . The notation  $[a]^+$  refers to  $\max\{a, 0\}$ . The set of integers ranging from  $a$  to  $b$  is denoted  $[a..b]$ . Given a set of  $n$  rvs  $\{B_i(t)\}_{i \in [1..n]}$ ,  $\vec{B}(t)$  denotes the vector  $(B_1(t), \dots, B_n(t))$  and  $\vec{B}$  denotes the stochastic process  $\{\vec{B}(t), t \geq 0\}$ .

## 5 Centralized Repair Systems

In this section, we address the performance of P2PSS using the centralized-recovery scheme.

We will focus on a single block of data  $D$ , and pay attention only to peers storing fragments of this block. At any time  $t$ , the state of a block  $D$  can be described by both the number of fragments that are available for download and the state of the recovery process. When triggered, the recovery process goes first through a “download phase” (fragments are downloaded from connected peers to the central authority) then through an “upload phase” (fragments are uploaded to new peers from the central authority).

More formally, we introduce  $n$ -dimensional vectors  $\vec{X}_c(t)$ ,  $\vec{Y}_c(t)$ ,  $\vec{Z}_c(t)$ ,  $\vec{U}_c(t)$ , and  $\vec{V}_c(t)$ , where  $n$  is the number of phases of the hyper-exponential distribution of peers on-times durations, and a  $5n$ -dimensional vector  $\vec{W}_c(t) = (\vec{X}_c(t), \vec{Y}_c(t), \vec{Z}_c(t), \vec{U}_c(t), \vec{V}_c(t))$ . Vectors  $\vec{Y}_c(t)$  and  $\vec{Z}_c(t)$  describe the download phase of the recovery process whereas  $\vec{U}_c(t)$  and  $\vec{V}_c(t)$  describe its upload phase. The formal definition of these vectors is as follows:

- $\vec{X}_c(t) := (X_{c,1}(t), \dots, X_{c,n}(t))$  where  $X_{c,l}(t)$  is a  $[0..s+r]$ -valued rv denoting the number of fragments of  $D$  stored on peers that are in phase  $l$  at time  $t$ .

- $\vec{Y}_c(t) := (Y_{c,1}(t), \dots, Y_{c,n}(t))$  where  $Y_{c,l}(t)$  is a  $[0..s-1]$ -valued rv denoting the number of fragments of  $D$  being downloaded at time  $t$  to the central authority from peers in phase  $l$  (one fragment per peer).
- $\vec{Z}_c(t) := (Z_{c,1}(t), \dots, Z_{c,n}(t))$  where  $Z_{c,l}(t)$  is a  $[0..s]$ -valued rv denoting the number of fragments of  $D$  hold at time  $t$  by the central authority and whose download was done from peers in phase  $l$  (one fragment per peer). Observe that these peers may have left the system by time  $t$ .
- $\vec{U}_c(t) := (U_{c,1}(t), \dots, U_{c,n}(t))$  where  $U_{c,l}(t)$  is a  $[0..s+r-1]$ -valued rv denoting the number of (reconstructed) fragments of  $D$  being uploaded at time  $t$  from the central authority to new peers that are in phase  $l$  (one fragment per peer).
- $\vec{V}_c(t) := (V_{c,1}(t), \dots, V_{c,n}(t))$  where  $V_{c,l}(t)$  is a  $[0..s+r-1]$ -valued rv denoting the number of (reconstructed) fragments of  $D$  whose upload from the central authority to new peers that are in phase  $l$  has been completed at time  $t$  (one fragment per peer).

Given the above definitions, we necessarily have  $Y_{c,l}(t) \leq X_{c,l}(t)$  for  $l \in [1..n]$  at any time  $t$ . The number of fragments of  $D$  that are available for download at time  $t$  is given by  $S(\vec{X}_c(t))$  (recall the definition of the function  $S$  in Section 3). Given that  $s$  fragments of  $D$  need to be downloaded to the central authority during the download phase of the recovery process, we will have (during this phase)  $S(\vec{Y}_c(t)) + S(\vec{Z}_c(t)) = s$ , such that  $S(\vec{Y}_c(t)), S(\vec{Z}_c(t)) \in [1..s-1]$ . Once the download phase is completed, the central authority will reconstruct at once all missing fragments, that is  $s+r-S(\vec{X}_c(t))$ . Therefore, during the upload phase, we have  $S(\vec{U}_c(t)) + S(\vec{V}_c(t)) = s+r-S(\vec{X}_c(t))$ . Observe that, once the download phase is completed, the number of available fragments,  $S(\vec{X}_c(t))$ , may well decrease to 0 with peers all leaving the system. In such a situation, the central authority will reconstruct  $s+r$  fragments of  $D$ . As soon as the download phase is completed  $\vec{Y}_c(t) = \vec{0}$  and  $S(\vec{Z}_c(t)) = s$ . The end of the upload phase is also the end of the recovery process. We will then have  $\vec{Y}_c(t) = \vec{Z}_c(t) = \vec{U}_c(t) = \vec{V}_c(t) = \vec{0}$  until the recovery process is again triggered.

According to the terminology introduced in Section 3, at time  $t$ , data  $D$  is *available* if  $S(\vec{X}_c(t)) \geq s$ , regardless of the state of the recovery process. It is *unavailable* if  $S(\vec{X}_c(t)) < s$  but  $S(\vec{Z}_c(t))$ —the number of fragments hold by the central authority—is larger than  $s - S(\vec{X}_c(t))$  and at least  $s - S(\vec{X}_c(t))$  fragments out of  $S(\vec{Z}_c(t))$  are different from those  $S(\vec{X}_c(t))$  fragments available on peers. Otherwise,  $D$  is considered to be *lost*. The latter situation will be modeled by a single state  $a$ .

If a recovery process is ongoing, the exact number of *distinct* fragments of  $D$  that are in the system—counting both those that are available and those hold by the central authority—may be unknown due to peers churn. However, we are able to find a lower bound on it, namely,

$$b(\vec{X}_c(t), \vec{Y}_c(t), \vec{Z}_c(t)) := \sum_{l=1}^n \max\{X_{c,l}(t), Y_{c,l}(t) + Z_{c,l}(t)\}.$$



In fact, the uncertainty about the number of distinct fragments is a result of peers churn. That said, this bound is very tight and most often gives the exact number of distinct fragments since peers churn occurs at a much larger time-scale than a fragment download. In our modeling, we consider an unavailable data  $D$  to become lost when the bound  $b$  takes a value smaller than  $s$ . Observe that, if the recovery process is not triggered, then  $b(\vec{X}_c(t), \vec{0}, \vec{0}) = S(\vec{X}_c(t))$  gives the exact number of distinct fragments.

The system state at time  $t$  can be represented by the  $5n$ -dimensional vector  $\vec{W}_c(t)$ . Thanks to the assumptions made in Section 3, the multi-dimensional process  $\vec{W}_c := \{\vec{W}_c(t), t \geq 0\}$  is an absorbing homogeneous continuous-time Markov chain (CTMC) with a set of transient states  $\mathcal{T}_c$  representing the situations when  $D$  is either available or unavailable and a single absorbing state  $a$  representing the situation when  $D$  is lost. As writing  $\mathcal{T}_c$  is tedious, we will simply say that  $\mathcal{T}_c$  is a subset of  $[0..s+r]^n \times [0..s-1]^n \times [0..s]^n \times [0..s+r-1]^n \times [0..s+r-1]^n$ . The elements of  $\mathcal{T}_c$  must verify the constraints mentioned above.

Without loss of generality, we assume that  $S(\vec{X}_c(0)) \geq s$ . The infinitesimal generator has the following canonical form

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

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  $\vec{w}_c \in \mathcal{T}_c$  and the absorbing state  $a$ . 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  $\vec{R}_c$  are, for  $S(\vec{y}_c) \in [1..S(\vec{x}_c)]$  and  $S(\vec{z}_c) = s - S(\vec{y}_c)$ ,

$$\begin{aligned} r_c(\vec{x}_c, \vec{0}, \vec{0}, \vec{0}, \vec{0}) &= \sum_{l=1}^n x_{c,l} \mu_l, & \text{for } S(\vec{x}_c) = s. \\ r_c(\vec{x}_c, \vec{y}_c, \vec{z}_c, \vec{0}, \vec{0}) &= \sum_{l=1}^n y_{c,l} \mu_l \cdot \mathbb{1}\{b(\vec{x}_c, \vec{y}_c, \vec{z}_c) = s\}, & \text{for } S(\vec{x}_c) \in [1..s]. \end{aligned}$$

Let us proceed to the definition of the non-zero elements of  $\vec{Q}_c$ .

#### The case when a peer leaves the system

There are seven different situations in this case. In the first situation, either the recovery process has not been triggered or it has but no download has been completed yet. In both the second and third situations, the download phase of the recovery process is ongoing and at least one download is completed. However, in the second situation, the departing peer does not affect the recovery process (either it was not involved in it or its fragment download is completed), unlike what happens in the third situation. In the third situation, a fragment download is interrupted due to the peer's departure. The central authority will then immediately start downloading a fragment from another available peer that is uniformly

selected among all available peers not currently involved in the recovery process. The fourth situation arises when a peer leaves the system at the end of the download phase. The fifth situation occurs when an available fragment becomes unavailable during the upload phase. The sixth situation occurs when a peer, to which the central authority is uploading a fragment, leaves the system. The last situation arises because of a departure of a peer to which the central authority has completely uploaded a reconstructed fragment. Note that the uploaded fragment was not yet integrated in the available fragments. This is caused by the fact that the central authority updates the database recording fragments locations as soon as all uploads terminate. To overcome any departure or failure that occurs in the context of one of the last three situations, the central authority has then to upload again the given fragment to a new peer. A new selected peer would be in phase  $m$  with probability  $R(m)$  for  $m \in [1..n]$ . The elements of  $\vec{Q}_c$  corresponding to these seven situations are, for  $l \in [1..n]$  and  $m \in [1..n]$ ,

$$\begin{aligned}
& q_c((\vec{x}_c, \vec{0}, \vec{0}, \vec{0}, \vec{0}), (\vec{x}_c - \vec{e}_n^l, \vec{0}, \vec{0}, \vec{0}, \vec{0})) = x_{c,l}\mu_l, \\
& \quad \text{for } S(\vec{x}_c) \in [s+1..s+r]. \\
& q_c((\vec{x}_c, \vec{y}_c, \vec{z}_c, \vec{0}, \vec{0}), (\vec{x}_c - \vec{e}_n^l, \vec{y}_c, \vec{z}_c, \vec{0}, \vec{0})) = [x_{c,l} - y_{c,l}]^+ \mu_l, \\
& \quad \text{for } S(\vec{x}_c) \in [s..s+r-1], \quad S(\vec{y}_c) \in [1..s-1], \quad S(\vec{z}_c) = s - S(\vec{y}_c); \\
& \quad \text{or } S(\vec{x}_c) \in [2..s-1], \quad S(\vec{y}_c) \in [1..S(\vec{x}_c)-1], \quad S(\vec{z}_c) = s - S(\vec{y}_c). \\
& q_c((\vec{x}_c, \vec{y}_c, \vec{z}_c, \vec{0}, \vec{0}), (\vec{x}_c - \vec{e}_n^l, \vec{y}_c - \vec{e}_n^l + \vec{e}_n^m, \vec{z}_c, \vec{0}, \vec{0})) = \frac{y_{c,l}\mu_l[x_{c,m} - y_{c,m} - z_{c,m}]^+}{\sum_{i=1}^n [x_{c,i} - y_{c,i} - z_{c,i}]^+}, \\
& \quad \text{for } S(\vec{x}_c) \in [s..s+r-1], \quad S(\vec{y}_c) \in [1..s-1], \quad S(\vec{z}_c) = s - S(\vec{y}_c); \\
& \quad \text{or } S(\vec{x}_c) \in [2..s-1], \quad S(\vec{y}_c) \in [1..S(\vec{x}_c)-1], \quad S(\vec{z}_c) = s - S(\vec{y}_c). \\
\\
& q_c((\vec{x}_c, \vec{0}, \vec{z}_c, \vec{0}, \vec{0}), (\vec{x}_c - \vec{e}_n^l, \vec{0}, \vec{z}_c, \vec{0}, \vec{0})) = x_{c,l}\mu_l, \\
& \quad \text{for } S(\vec{x}_c) \in [1..s+r-1], \quad S(\vec{z}_c) = s. \\
& q_c((\vec{x}_c, \vec{0}, \vec{z}_c, \vec{u}_c, \vec{v}_c), (\vec{x}_c - \vec{e}_n^l, \vec{0}, \vec{z}_c, \vec{u}_c + \vec{e}_n^m, \vec{v}_c)) = x_{c,l}\mu_l R(m), \\
& \quad \text{for } S(\vec{x}_c) \in [1..s+r-2], \quad S(\vec{z}_c) = s, \quad S(\vec{u}_c) \in [1..s+r-S(\vec{x}_c)-1], \\
& \quad \quad S(\vec{v}_c) = s+r-S(\vec{x}_c)-S(\vec{u}_c). \\
& q_c((\vec{x}_c, \vec{0}, \vec{z}_c, \vec{u}_c, \vec{v}_c), (\vec{x}_c, \vec{0}, \vec{z}_c, \vec{u}_c - \vec{e}_n^l + \vec{e}_n^m, \vec{v}_c)) = u_{c,l}\mu_l R(m), \\
& \quad \text{for } S(\vec{x}_c) \in [1..s+r-2], \quad S(\vec{z}_c) = s, \quad S(\vec{u}_c) \in [1..s+r-S(\vec{x}_c)-1], \\
& \quad \quad S(\vec{v}_c) = s+r-S(\vec{x}_c)-S(\vec{u}_c), \quad l \neq m. \\
& q_c((\vec{x}_c, \vec{0}, \vec{z}_c, \vec{u}_c, \vec{v}_c), (\vec{x}_c, \vec{0}, \vec{z}_c, \vec{u}_c + \vec{e}_n^m, \vec{v}_c - \vec{e}_n^l)) = v_{c,l}\mu_l R(m), \\
& \quad \text{for } S(\vec{x}_c) \in [1..s+r-2], \quad S(\vec{z}_c) = s, \quad S(\vec{u}_c) \in [1..s+r-S(\vec{x}_c)-1], \\
& \quad \quad S(\vec{v}_c) = s+r-S(\vec{x}_c)-S(\vec{u}_c).
\end{aligned}$$

### The case when a peer rejoins the system

Recall that the system keeps trace of only the latest known location of each fragment. As such, 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. In the following,

only relevant reconnections are considered. For instance, when the recovery process is in its upload phase, any peer that rejoins the system does not affect the system state since all fragments have been reconstructed and are being uploaded to their new locations.

There are three situations where reconnections may be relevant. In the first, either the recovery process has not been triggered or it has but no download has been completed yet. In both the second and third situations, the download phase of the recovery process is ongoing and at least one download is completed. However, in the third situation, there is only one missing fragment, so when the peer storing the missing fragments rejoins the system, the recovery process aborts.

The elements of  $\vec{Q}_c$  corresponding to these three situations are, for  $l \in [1..n]$  and  $S(\vec{z}_c) = s - S(\vec{y}_c)$

$$\begin{aligned} q_c((\vec{x}_c, \vec{0}, \vec{0}, \vec{0}, \vec{0}), (\vec{x}_c + \vec{e}_n^l, \vec{0}, \vec{0}, \vec{0}, \vec{0})) &= p_l(s + r - S(\vec{x}_c))p\lambda, \\ &\text{for } S(\vec{x}_c) \in [s..s + r - 1]. \\ q_c((\vec{x}_c, \vec{y}_c, \vec{z}_c, \vec{0}, \vec{0}), (\vec{x}_c + \vec{e}_n^l, \vec{y}_c, \vec{z}_c, \vec{0}, \vec{0})) &= p_l(s + r - S(\vec{x}_c))p\lambda, \\ &\text{for } S(\vec{x}_c) \in [s..s + r - 2], \quad S(\vec{y}_c) \in [1..s - 1]; \\ &\text{or } S(\vec{x}_c) \in [1..s - 1], \quad S(\vec{y}_c) \in [1..S(\vec{x}_c)]. \\ q_c((\vec{x}_c, \vec{y}_c, \vec{z}_c, \vec{0}, \vec{0}), (\vec{x}_c + \vec{e}_n^l, \vec{0}, \vec{0}, \vec{0}, \vec{0})) &= p_l p\lambda, \\ &\text{for } S(\vec{x}_c) = s + r - 1, \quad S(\vec{y}_c) \in [1..s - 1]. \end{aligned}$$

#### The case when one download is completed during the recovery process

When a recovery process is initiated, the system state verifies  $S(\vec{x}_c) \in [s..s + r - k]$  and  $\vec{y}_c = \vec{z}_c = \vec{u}_c = \vec{v}_c = \vec{0}$ . The central authority selects  $s$  peers out of the  $S(\vec{x}_c)$  peers that are connected to the system and initiates a fragment download from each. Among the  $s$  peers that are selected,  $i_l$  out of  $s$  would be in phase  $l$ , for  $l \in [1..n]$ . Let  $\vec{i} = (i_1, \dots, i_n)$ . We naturally have  $0 \leq i_l \leq x_{c,l}$ , for  $l \in [1..n]$ , and  $S(\vec{i}) = s$ . This selection occurs with probability

$$g(\vec{i}, \vec{x}_c) := \frac{\prod_{l=1}^n \binom{x_{c,l}}{i_l}}{\binom{S(\vec{x}_c)}{s}}.$$

The probability that the first download to be completed out of  $s$  was from a peer in phase  $l$  is equal to  $f_l(\vec{i}) = i_l/s$  (recall the definition of  $f$  in Section 3). Similarly, when the number of ongoing downloads is  $\vec{y}_c$ , the probability that the first download to be completed out of  $S(\vec{y}_c)$  was from a peer in phase  $l$  is equal to  $f_l(\vec{y}_c) = y_{c,l}/S(\vec{y}_c)$ .

The two possible transition rates in such situations are, for  $l \in [1..n]$ ,  $m \in [1..n]$  and  $S(\vec{z}_c) = s - S(\vec{y}_c)$ ,

$$\begin{aligned} q_c((\vec{x}_c, \vec{0}, \vec{0}, \vec{0}, \vec{0}), (\vec{x}_c, \vec{i} - \vec{e}_n^l, \vec{e}_n^l, \vec{0}, \vec{0})) &= s\alpha g(\vec{i}, \vec{x}_c) f_l(\vec{i}), \\ &\text{for } S(\vec{x}_c) \in [s..s + r - k], \quad i_m \in [0..x_{c,m}], \quad S(\vec{i}) = s. \\ q_c((\vec{x}_c, \vec{y}_c, \vec{z}_c, \vec{0}, \vec{0}), (\vec{x}_c, \vec{y}_c - \vec{e}_n^l, \vec{z}_c + \vec{e}_n^l, \vec{0}, \vec{0})) &= S(\vec{y}_c)\alpha f_l(\vec{y}_c), \\ &\text{for } S(\vec{x}_c) \in [s..s + r - 1], \quad S(\vec{y}_c) \in [1..s - 1]; \\ &\text{or } S(\vec{x}_c) \in [1..s - 1], \quad S(\vec{y}_c) \in [1..S(\vec{x}_c)]. \end{aligned}$$

#### The case when one upload is completed during the recovery process

When the download phase is completed, the system state verifies  $S(\vec{z}_c) = s$  and  $\vec{y}_c = \vec{u}_c = \vec{v}_c = \vec{0}$ . The central authority selects  $s + r - S(\vec{x}_c)$  new peers that are connected to the system and initiates a (reconstructed) fragment upload to each. Among the peers that are selected,  $i_l$  out of  $s + r - S(\vec{x}_c)$  would be in phase  $l$ , for  $l \in [1..n]$ . Let  $\vec{i} = (i_1, \dots, i_n)$ . We naturally have  $0 \leq i_l \leq s + r - S(\vec{x}_c)$ , for  $l \in [1..n]$ , and  $S(\vec{i}) = s + r - S(\vec{x}_c)$ . This selection occurs with probability

$$h(\vec{i}, \vec{x}_c) := \binom{s + r - S(\vec{x}_c)}{i_1, i_2, \dots, i_n} \prod_{l=1}^n R(l)^{i_l}$$

where the multinomial coefficient has been used. For  $l \in [1..n]$  and  $S(\vec{z}_c) = s$ , we can write

$$\begin{aligned} q_c((\vec{x}_c, \vec{0}, \vec{z}_c, \vec{0}, \vec{0}), (\vec{x}_c, \vec{0}, \vec{z}_c, \vec{i} - \vec{e}_n^l, \vec{e}_n^l)) &= S(\vec{i})\beta h(\vec{i}, \vec{x}_c) f_i(\vec{i}), \\ \text{for } S(\vec{x}_c) &\in [0..s + r - 2], \quad \vec{i} \in [0..s + r - S(\vec{x}_c)]^n, \quad S(\vec{i}) = s + r - S(\vec{x}_c). \\ q_c((\vec{x}_c, \vec{0}, \vec{z}_c, \vec{u}_c, \vec{v}_c), (\vec{x}_c, \vec{0}, \vec{z}_c, \vec{u}_c - \vec{e}_n^l, \vec{v}_c + \vec{e}_n^l)) &= S(\vec{u}_c)\beta f_i(\vec{u}_c), \\ \text{for } S(\vec{x}_c) &\in [0..s + r - 2], \quad S(\vec{u}_c) \in [2..s + r - S(\vec{x}_c) - 1], \\ S(\vec{v}_c) &= s + r - S(\vec{x}_c) - S(\vec{u}_c). \\ q_c((\vec{x}_c, \vec{0}, \vec{z}_c, \vec{e}_n^l, \vec{v}_c), (\vec{x}_c + \vec{v}_c + \vec{e}_n^l, \vec{0}, \vec{0}, \vec{0}, \vec{0})) &= \beta, \\ \text{for } S(\vec{x}_c) &\in [0..s + r - 2], \quad S(\vec{v}_c) = s + r - S(\vec{x}_c) - 1. \\ q_c((\vec{x}_c, \vec{0}, \vec{z}_c, \vec{0}, \vec{0}), (\vec{x}_c + \vec{e}_n^l, \vec{0}, \vec{0}, \vec{0}, \vec{0})) &= R(l)\beta, \\ \text{for } S(\vec{x}_c) &= s + r - 1. \end{aligned}$$

Note that  $\vec{Q}_c$  is not an infinitesimal generator since elements in some rows do not sum up to 0. Those rows correspond to the system states where only  $s$  distinct fragments are present in the system. The diagonal elements of  $\vec{Q}_c$  are

$$q_c(\vec{w}_c, \vec{w}_c) = -r_c(\vec{w}_c) - \sum_{\vec{w}'_c \in \mathcal{T}_c - \{\vec{w}_c\}} q_c(\vec{w}_c, \vec{w}'_c), \quad \text{for } \vec{w}_c \in \mathcal{T}_c.$$

For illustration purposes, we depict in Fig. 1 some of the transitions of the absorbing CTMC when  $n = 2$ ,  $s = 3$ ,  $r = 1$ , and  $k = 1$ .

## 5.1 Data Lifetime

This section is devoted to the analysis of the lifetime of  $D$ . It will be convenient to introduce sets

$$\mathcal{E}_I := \{(\vec{x}_c, \vec{0}, \vec{0}, \vec{0}, \vec{0}) : \vec{x}_c \in [0..s + r]^n, S(\vec{x}_c) = I\} \text{ for } I \in [s..s + r].$$

The set  $\mathcal{E}_I$  consists of all states of the process  $\vec{W}_c$  in which the number of fragments of  $D$  currently available is equal to  $I$  and the recovery process either has not been triggered (for  $I \in [s + r - k + 1..s + r]$ ) or it has but no download has been completed yet (for  $I \in [s..s + r - k]$ ). For any  $I$ , the cardinal of  $\mathcal{E}_I$  is  $\binom{I+n-1}{n-1}$  (think of the possible selections of  $n - 1$  boxes in a row of  $I + n - 1$  boxes, so as to delimit  $n$  groups of boxes summing up to  $I$ ).

Introduce  $T_c(\mathcal{E}_I) := \inf\{t > 0 : \vec{W}_c(t) = a | \vec{W}_c(0) \in \mathcal{E}_I\}$ , the time until absorption in state  $a$ —or equivalently the time until  $D$  is lost—given that the initial number of fragments of  $D$  available in the system is equal to  $I$ . In the following,  $T_c(\mathcal{E}_I)$  will be referred to as the *conditional block lifetime*. We are interested in the conditional probability distribution function,  $P(T_c(\mathcal{E}_I) \leq t)$ , and the conditional expectation,  $E[T_c(\mathcal{E}_I)]$ , given that  $\vec{W}_c(0) \in \mathcal{E}_I$  for  $I \in [s..s+r]$ .

From the theory of absorbing Markov chains, we can compute  $P(T_c(\{\vec{w}_c\}) \leq t)$  where  $T_c^h(\{\vec{w}_c\})$  is the time until absorption in state  $a$  given that the system initiates in state  $\vec{w}_c \in \mathcal{T}_c$ . We know that (e.g. [19, Lemma 2.2])

$$P(T_c(\{\vec{w}_c\}) \leq t) = 1 - \vec{e}_{|\mathcal{T}_c|}^{\text{ind}(\vec{w}_c)} \cdot \exp(t\vec{Q}_c) \cdot \vec{1}_{|\mathcal{T}_c|}, \quad t > 0, \vec{w}_c \in \mathcal{T}_c \quad (2)$$

where  $\text{ind}(\vec{w}_c)$  refers to the index of state  $\vec{w}_c$  in the matrix  $\vec{Q}_c$ . Definitions of vectors  $\vec{e}_j^i$  and  $\vec{1}_j$  were given at the end of Section 3. Observe that the term  $\vec{e}_{|\mathcal{T}_c|}^{\text{ind}(\vec{w}_c)} \cdot \exp(t\vec{Q}_c) \cdot \vec{1}_{|\mathcal{T}_c|}$  in the right-hand side of (2) is nothing but the summation of all  $|\mathcal{T}_c|$  elements in row  $\text{ind}(\vec{w}_c)$  of matrix  $\exp(t\vec{Q}_c)$ .

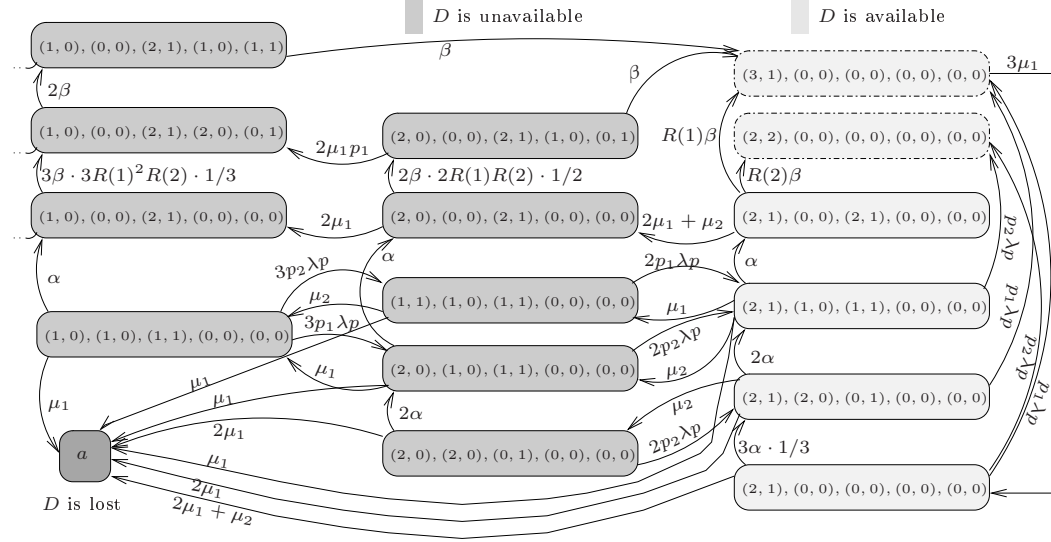


Figure 1: Some transitions of the Markov chain  $\vec{W}_c$  when  $n = 2, s = 3, r = 1$ , and  $k = 1$ .

Let  $\pi_{\vec{x}_c}$  denote the probability that the system starts in state  $\vec{w}_c = (\vec{x}_c, \vec{0}, \vec{0}, \vec{0}, \vec{0}) \in \mathcal{E}_I$  at time 0 given that  $\vec{W}_c(0) \in \mathcal{E}_I$ . We can write

$$\pi_{\vec{x}_c} := P\left(\vec{W}_c(0) = \vec{w}_c \in \mathcal{E}_I | \vec{W}_c(0) \in \mathcal{E}_I\right) = \binom{I}{x_{c,1}, \dots, x_{c,n}} \prod_{l=1}^n R(l)^{x_{c,l}}. \quad (3)$$

Clearly  $\sum_{\vec{w}_c \in \mathcal{E}_I} \pi_{\vec{x}_c} = 1$  for  $I \in [s..s+r]$ . Using (2) and (3) and the total probability theorem yields, for  $I \in [s..s+r]$ ,

$$\begin{aligned} P(T_c(\mathcal{E}_I) \leq t) &= \sum_{\vec{w}_c \in \mathcal{E}_I} \pi_{\vec{x}_c} P(T_c(\{\vec{w}_c\}) \leq t) \\ &= 1 - \sum_{\vec{w}_c \in \mathcal{E}_I} \pi_{\vec{x}_c} \vec{e}_{|\mathcal{T}_c|}^{\text{ind}(\vec{w}_c)} \cdot \exp\left(t\vec{Q}_c\right) \cdot \vec{1}_{|\mathcal{T}_c|}, \quad t > 0. \end{aligned} \quad (4)$$

We know from [19, p. 46] that the expected time until absorption given that the  $\vec{W}_c(0) = \vec{w}_c \in \mathcal{T}_c$  can be written as

$$\mathbb{E}[T_c(\{\vec{w}_c\})] = -\vec{e}_{|\mathcal{T}_c|}^{\text{ind}(\vec{w}_c)} \cdot \left(\vec{Q}_c\right)^{-1} \cdot \vec{1}_{|\mathcal{T}_c|}, \quad \vec{w}_c \in \mathcal{T}_c,$$

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 [19, p. 45]. The conditional expectation of  $T_c(\mathcal{E}_I)$  is then (recall that the elements of  $\mathcal{E}_I$  are of the form  $(\vec{x}_c, \vec{0}, \vec{0}, \vec{0}, \vec{0})$ )

$$\begin{aligned} \mathbb{E}[T_c(\mathcal{E}_I)] &= \sum_{\vec{w}_c \in \mathcal{E}_I} \pi_{\vec{x}_c} \mathbb{E}[T_c(\{\vec{w}_c\})] \\ &= - \sum_{\vec{w}_c \in \mathcal{E}_I} \pi_{\vec{x}_c} \vec{e}_{|\mathcal{T}_c|}^{\text{ind}(\vec{w}_c)} \cdot \left(\vec{Q}_c\right)^{-1} \cdot \vec{1}_{|\mathcal{T}_c|}, \quad \text{for } I \in [s..s+r]. \end{aligned} \quad (5)$$

## 5.2 Data Availability

In this section we introduce different metrics to quantify the availability of  $D$ . But first, we will study the time during which  $J$  fragments of  $D$  are available in the system given that there were initially  $I$  fragments. To formalize this measure, we introduce the following subsets of  $\mathcal{T}_c$ , for  $J \in [0..s+r]$ ,

$$\mathcal{F}_J := \{(\vec{x}_c, \vec{y}_c, \vec{z}_c, \vec{u}_c, \vec{v}_c) \in \mathcal{T}_c : S(\vec{x}_c) = J\}$$

The set  $\mathcal{F}_J$  consists of all states of process  $\vec{W}_c$  in which the number of fragments of  $D$  currently available is equal to  $J$ , regardless of the state of the recovery process. The subsets  $\mathcal{F}_J$  form a partition of  $\mathcal{T}_c$ . We may define now

$$T_c(\mathcal{E}_I, \mathcal{F}_J) := \int_0^{T_c(\mathcal{E}_I)} \mathbb{1}\left\{\vec{W}_c(t) \in \mathcal{F}_J | \vec{W}_c(0) \in \mathcal{E}_I\right\} dt.$$

$T_c(\mathcal{E}_I, \mathcal{F}_J)$  is the total time spent by the CTMC in the set  $\mathcal{F}_J$  before being absorbed in state  $a$ , given that  $\vec{W}_c(0) \in \mathcal{E}_I$ . Similarly,  $T_c(\{\vec{w}_c\}, \{\vec{w}'_c\})$  is the total time spent by the CTMC in state  $\vec{w}'_c$  before being absorbed in state  $a$ , given that  $\vec{W}_c(0) = \vec{w}_c$ . We know from [14, p. 419] that

$$\mathbb{E}[T_c(\{\vec{w}_c\}, \{\vec{w}'_c\})] = -\vec{e}_{|\mathcal{T}_c|}^{\text{ind}(\vec{w}_c)} \cdot \left(\vec{Q}_c\right)^{-1} \cdot {}^t\vec{e}_{|\mathcal{T}_c|}^{\text{ind}(\vec{w}'_c)}, \quad \vec{w}_c, \vec{w}'_c \in \mathcal{T}_c \quad (6)$$

where  ${}^t\vec{y}$  denotes the transpose of a given vector  $\vec{y}$ . In other words, the expectation  $\mathbb{E}[T_c(\{\vec{w}_c\}, \{\vec{w}'_c\})]$  is the entry of matrix  $\left(-\vec{Q}_c\right)^{-1}$  at row  $\text{ind}(\vec{w}_c)$  and column  $\text{ind}(\vec{w}'_c)$ . Using (3) and (6), we derive for  $I \in [s..s+r]$  and  $J \in [0..s+r]$

$$\begin{aligned} \mathbb{E}[T_c(\mathcal{E}_I, \mathcal{F}_J)] &= \sum_{\vec{w}'_c \in \mathcal{F}_J} \mathbb{E}[T_c(\mathcal{E}_I, \{\vec{w}'_c\})] \\ &= \sum_{\vec{w}_c \in \mathcal{E}_I} \sum_{\vec{w}'_c \in \mathcal{F}_J} \pi_{\vec{x}_c} \mathbb{E}[T_c(\{\vec{w}_c\}, \{\vec{w}'_c\})] \\ &= - \sum_{\vec{w}_c \in \mathcal{E}_I} \sum_{\vec{w}'_c \in \mathcal{F}_J} \pi_{\vec{x}_c} \vec{e}_{|\mathcal{T}_c|}^{\text{ind}(\vec{w}_c)} \cdot \left(\vec{Q}_c\right)^{-1} \cdot {}^t\vec{e}_{|\mathcal{T}_c|}^{\text{ind}(\vec{w}'_c)}. \end{aligned} \quad (7)$$

We are now in position of introducing two availability metrics. The first metric, defined as

$$M_{c,1}(\mathcal{E}_I) := \mathbb{E} \left[ \sum_{J=0}^{s+r} J \frac{T_c(\mathcal{E}_I, \mathcal{F}_J)}{T_c(\mathcal{E}_I)} \right], \quad \text{where } I \in [s..s+r],$$

can be interpreted as the expected number of fragments of  $D$  that are available for download—as long as  $D$  is not lost—given that  $I$  fragments are initially available. A second metric is

$$M_{c,2}(\mathcal{E}_I, m) := \mathbb{E} \left[ \sum_{J=m}^{s+r} \frac{T_c(\mathcal{E}_I, \mathcal{F}_J)}{T_c(\mathcal{E}_I)} \right], \quad \text{where } I \in [s..s+r],$$

that we can interpret as the fraction of the lifetime of  $D$  when at least  $m$  fragments are available for download, given that  $I$  fragments are initially available. For instance,  $M_{c,2}(\mathcal{E}_{s+r}, s)$  is the proportion of time when data  $D$  is *available* for users, given that  $s+r$  fragments of  $D$  are initially available for download.

The expectations involved in the computation of the availability metrics are difficult to find in closed-form. Therefore, we resort to using the following approximation

$$\mathbb{E} \left[ \frac{T_c(\mathcal{E}_I, \mathcal{F}_J)}{T_c(\mathcal{E}_I)} \right] \approx \frac{\mathbb{E}[T_c(\mathcal{E}_I, \mathcal{F}_J)]}{\mathbb{E}[T_c(\mathcal{E}_I)]}, \quad (8)$$

where the terms in the right-hand side have been derived in (7) and (5). We will come back to this approximation in Section 7. With this approximation in mind, the two availability

metrics become

$$M_{c,1}(\mathcal{E}_I) = \sum_{J=0}^{s+r} J \frac{\mathbb{E}[T_c(\mathcal{E}_I, \mathcal{F}_J)]}{\mathbb{E}[T_c(\mathcal{E}_I)]}, \quad \text{where } I \in [s..s+r], \quad (9)$$

$$M_{c,2}(\mathcal{E}_I, m) = \sum_{J=m}^{s+r} \frac{\mathbb{E}[T_c(\mathcal{E}_I, \mathcal{F}_J)]}{\mathbb{E}[T_c(\mathcal{E}_I)]}, \quad \text{where } I \in [s..s+r]. \quad (10)$$

## 6 Distributed Repair Systems

In this section, we model P2P storage systems that implement a distributed recovery mechanism, as described in Section 3. The system at time  $t$  can be described by the number of fragments that are available for download and the state of the recovery process. Unlike the centralized case, the distributed recovery process consists of only a download phase at the end of which the secure agent running on the new peer reconstructs a *single* fragment and stores it on the peer's disk.

To model the system, we introduce  $n$ -dimensional vectors  $\vec{X}_d(t)$ ,  $\vec{Y}_d(t)$ ,  $\vec{Z}_d(t)$  and a  $3n$ -dimensional vector  $\vec{W}_d(t) = (\vec{X}_d(t), \vec{Y}_d(t), \vec{Z}_d(t))$ . Vectors  $\vec{Y}_d(t)$  and  $\vec{Z}_d(t)$  describe the recovery process. The formal definition of these vectors is as follows:

- $\vec{X}_d(t) := (X_{d,1}(t), \dots, X_{d,n}(t))$  where  $X_{d,l}(t)$  is a  $[0..s+r]$ -valued rv denoting the number of fragments of  $D$  stored on peers that are in phase  $l$  at time  $t$ .  $\vec{X}_d(t)$  must verify  $S(\vec{X}_d(t)) \in [s-1..s+r]$ .
- $\vec{Y}_d(t) := (Y_{d,1}(t), \dots, Y_{d,n}(t))$  where  $Y_{d,l}(t)$  is a  $[0..s-1]$ -valued rv denoting the number of fragments of  $D$  being downloaded at time  $t$  to the secure agent from peers in phase  $l$  (one fragment per peer).
- $\vec{Z}_d(t) := (Z_{d,1}(t), \dots, Z_{d,n}(t))$  where  $Z_{d,l}(t)$  is a  $[0..s-1]$ -valued rv denoting the number of fragments of  $D$  hold at time  $t$  by the secure agent and whose download was done from peers in phase  $l$  (one fragment per peer). Observe that these peers may have left the system by time  $t$ .

As in the centralized case,  $Y_{d,l}(t) \leq X_{d,l}(t)$  for  $l \in [1..n]$  at any time  $t$ . The number of fragments of  $D$  that are available for download at time  $t$  is given by  $S(\vec{X}_d(t))$ . During the recovery process,  $S(\vec{Y}_d(t)) + S(\vec{Z}_d(t)) = s$ , such that  $S(\vec{Y}_d(t)), S(\vec{Z}_d(t)) \in [1..s-1]$ . Because the distributed scheme repairs fragments only one at a time, we have  $S(\vec{X}_d(t)) \in [s-1..s+r]$ . The end of the download phase is also the end of the recovery process. We will then have  $\vec{Y}_d(t) = \vec{Z}_d(t) = \vec{0}$  until the recovery process is again triggered.

$D$  is available when  $S(\vec{X}_d(t)) \geq s$ , unavailable when  $S(\vec{X}_d(t)) = s-1$  and  $b(\vec{X}_d(t), \vec{Y}_d(t), \vec{Z}_d(t)) \geq s$  (recall the lower bound on the number of distinct fragments in the system introduced in Section 5) and lost otherwise (situation modeled by a single absorbing state  $a$ ).



According to the description and assumptions listed in Section 3, the state of data  $D$  at time  $t$  can be represented by  $\vec{W}_d(t)$  and the multi-dimensional process  $\vec{W}_d := \{\vec{W}_d(t), t \geq 0\}$  is an absorbing homogeneous CTMC with a single absorbing state  $a$ . The set of transient states  $\mathcal{T}_d$  is the set of elements of  $[0..s+r]^n \times [0..s-1]^n \times [0..s-1]^n$  that verify the constraints mentioned above.

The analysis of the absorbing Markov chain  $\vec{W}_d$  that takes values in  $\mathcal{T}_d \cup \{a\}$  is very similar to the analysis of  $\vec{W}_c$  in Section 5, 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  $\vec{R}_d$  are, for  $S(\vec{y}_d) \in [1..s-1]$  and  $S(\vec{z}_d) = s - S(\vec{y}_d)$ ,

$$\begin{aligned} r_d(\vec{x}_d, \vec{0}, \vec{0}) &= \sum_{l=1}^n x_{d,l} \mu_l, & \text{for } S(\vec{x}_d) = s. \\ r_d(\vec{x}_d, \vec{y}_d, \vec{z}_d) &= \sum_{l=1}^n x_{d,l} \mu_l, & \text{for } S(\vec{x}_d) = s-1. \\ r_d(\vec{x}_d, \vec{y}_d, \vec{z}_d) &= \sum_{l=1}^n y_{d,l} \mu_l \cdot \mathbb{1}\{b(\vec{x}_d, \vec{y}_d, \vec{z}_d) = s\}, & \text{for } S(\vec{x}_d) = s. \end{aligned}$$

We next write the non-zero elements of  $\vec{Q}_d$ .

#### The case when a peer leaves the system

There are three situations in this case. In the first situation, either the recovery process has not been triggered or it has but no download has been completed yet. In the other two situations, the recovery process is ongoing and at least one download is completed. In the second situation, the departing peer does not affect the recovery process, which is not the case of the third situation, where the secure agent must start downloading a fragment from another available peer that is uniformly selected among all available peers not currently involved in the recovery process. This peer would be in phase  $m$  with probability  $p_m$  for  $m \in [1..n]$ . The elements of  $\vec{Q}_d$  corresponding to these three situations are, for  $l \in [1..n]$ ,  $m \in [1..n]$ ,  $S(\vec{y}_d) \in [1..s-1]$  and  $S(\vec{z}_d) = s - S(\vec{y}_d)$ ,

$$\begin{aligned} q_d((\vec{x}_d, \vec{0}, \vec{0}), (\vec{x}_d - \vec{e}_n^l, \vec{0}, \vec{0})) &= x_{d,l} \mu_l, \\ &\text{for } S(\vec{x}_d) \in [s+1..s+r]. \\ q_d((\vec{x}_d, \vec{y}_d, \vec{z}_d), (\vec{x}_d - \vec{e}_n^l, \vec{y}_d, \vec{z}_d)) &= [x_{d,l} - y_{d,l}]^+ \mu_l, \\ &\text{for } S(\vec{x}_d) \in [s..s+r-1]. \\ q_d((\vec{x}_d, \vec{y}_d, \vec{z}_d), (\vec{x}_d - \vec{e}_n^l, \vec{y}_d - \vec{e}_n^l + \vec{e}_n^m, \vec{z}_d)) &= \frac{y_{d,l} \mu_l [x_{d,m} - y_{d,m} - z_{d,m}]^+}{\sum_{i=1}^n [x_{d,i} - y_{d,i} - z_{d,i}]^+}, \\ &\text{for } S(\vec{x}_d) \in [s..s+r-1]. \end{aligned}$$

#### The case when a peer rejoins the system

There are three situations in this case exactly like in Section 5. The elements of  $\vec{Q}_d$  corresponding to these three situations are, for  $l \in [1..n]$ ,  $S(\vec{y}_d) \in [1..s-1]$  and  $S(\vec{z}_d) =$

$$s - S(\vec{y}_d),$$

$$\begin{aligned} q_d((\vec{x}_d, \vec{0}, \vec{0}), (\vec{x}_d + \vec{e}_n^l, \vec{0}, \vec{0})) &= p_l(s + r - S(\vec{x}_d))p\lambda, & \text{for } S(\vec{x}_d) \in [s..s + r - 1]. \\ q_d((\vec{x}_d, \vec{y}_d, \vec{z}_d), (\vec{x}_d + \vec{e}_n^l, \vec{y}_d, \vec{z}_d)) &= p_l(s + r - S(\vec{x}_d))p\lambda, & \text{for } S(\vec{x}_d) \in [s - 1..s + r - 2]. \\ q_d((\vec{x}_d, \vec{y}_d, \vec{z}_d), (\vec{x}_d + \vec{e}_n^l, \vec{0}, \vec{0})) &= p_l p\lambda, & \text{for } S(\vec{x}_d) = s + r - 1. \end{aligned}$$

### The case when one download is completed during the recovery process

There are three situations in this case, following which download has been completed. If it is the first or any of the  $s - 2$  subsequent ones, then we obtain the two situations described in Section 5. The third situation occurs when the last download is completed, which is essentially the end of the recovery phase. The elements of  $\vec{Q}_d$  corresponding to these three situations are, for  $l \in [1..n]$  and  $m \in [1..n]$ ,

$$\begin{aligned} q_d((\vec{x}_d, \vec{0}, \vec{0}), (\vec{x}_d, \vec{i} - \vec{e}_n^l, \vec{e}_n^l)) &= s\alpha g(\vec{i}, \vec{x}_d) f_l(\vec{i}), \\ &\text{for } S(\vec{x}_d) \in [s..s + r - k], \quad i_l \in [0..x_{d,l}], \quad S(\vec{i}) = s. \\ q_d((\vec{x}_d, \vec{y}_d, \vec{z}_d), (\vec{x}_d, \vec{y}_d - \vec{e}_n^l, \vec{z}_d + \vec{e}_n^l)) &= S(\vec{y}_d)\alpha f_l(\vec{y}_d), \\ &\text{for } S(\vec{x}_d) \in [s - 1..s + r - 1], \quad S(\vec{y}_d) \in [2..s - 1], \quad S(\vec{z}_d) = s - S(\vec{y}_d). \\ q_d((\vec{x}_d, \vec{e}_n^m, \vec{z}_d), (\vec{x}_d + \vec{e}_n^l, \vec{0}, \vec{0})) &= R(l)\alpha, \\ &\text{for } S(\vec{x}_d) \in [s - 1..s + r - 1], \quad S(\vec{z}_d) = s - 1. \end{aligned}$$

And last :

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

For illustration purposes, we depict in Fig. 2 some of the transitions of the absorbing CTMC when  $n = 2$ ,  $s = 4$ ,  $r = 2$ , and  $k = 1$ .

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 5.  $P(T_d(\mathcal{E}_I) \leq t)$ ,  $E[T_d(\mathcal{E}_I)]$ ,  $E[T_d(\mathcal{E}_I, \mathcal{F}_J)]$ ,  $M_{d,1}(\mathcal{E}_I)$  and  $M_{d,2}(\mathcal{E}_I, m)$  are given in (4), (5), (7), (9) and (10) respectively, after replacing 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  $\vec{Q}_d$ .

## 7 Validation of (8)

In this section, we validate the approximation made in (8) which has been made to compute the two availability metrics in both models presented in this report. In order to do that, we need to simulate the Markov chain until absorption, and measure  $T(\mathcal{E}_I, \mathcal{F}_J)$  and  $T(\mathcal{E}_I)$  (we dropped the subscript as this applies to both centralized and distributed schemes). The sample mean of their ratio should then be compared to the ratio of the analytical expression. Each simulation scenario should be repeated many times in order to have a good estimation of the mean with a good confidence interval.

We decided to simulate the Markov chain  $\vec{W}_d$  as its state-space is smaller than that of  $\vec{W}_c$ . To reduce even more the state-space, we set  $n = 1$ . In other words, peers on-times durations are exponentially distributed. The environment that we simulated has the following characteristics: the expected off-time is  $1/\lambda = 1$  hour; the expected on-time is  $1/\mu = 3$  hour; the persistence probability is  $p = 0.7$ ; the original number of fragments of  $D$  is  $s = 8$ ; the fragment size is 1MB. We simulated a total of 10 different scenarios, each having different values of  $r$  and  $k$ . We have varied  $r$  from 1 to 4 and  $k$  from 1 to  $r$ . We have a total of 320 different instances over all scenarios. In order to obtain a maximum estimation error of about 1% with 97% confidence interval, we need to have over 150 sampled values. Hence, each scenario is simulated 150 times.

For each instance, we collect the 150 simulated values of the ratio  $T(\mathcal{E}_I, \mathcal{F}_J)/T(\mathcal{E}_I)$  and compute their average. This is the estimation of the left-hand side of (8) and will be considered as the “correct” value. The right-hand side of (8) is computed using the simplified versions of (5) and (7) that appeared in [10], where  $n = 1$  (after replacing the subscript “c” with the subscript “d”). This is the “approximate” value. We compute the relative error between the correct value and the approximate value. Having collected all values of the relative error from all 320 instances, we derive the empirical complementary cumulative distribution function of the relative error, as depicted in Fig. 3.

We have found that only 10% of the values are larger than  $0.9 \times 10^{-3}$  and, most importantly, the maximum value of the relative error is 0.0028. *We conclude that the approximation (8) is very good.* This approximation has been validated also in our previous work [1] under different assumptions. Even though we have not simulated the Markov chain  $\vec{W}_c$  presented in this report, we are convinced that the approximation will be equally good.

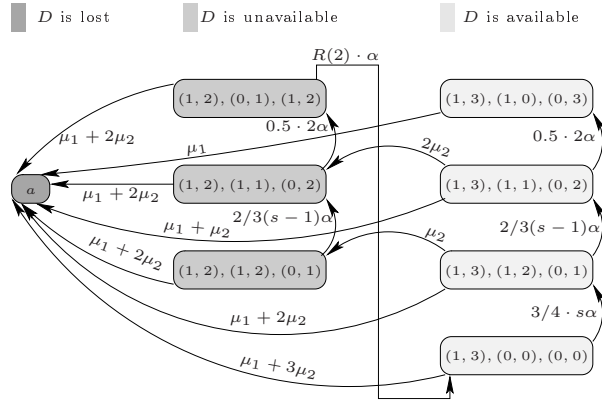


Figure 2: Some transition rates of the Markov chain  $\vec{W}_d$  when  $n = 2$ ,  $s = 4$ ,  $r = 2$ , and  $k = 1$ .

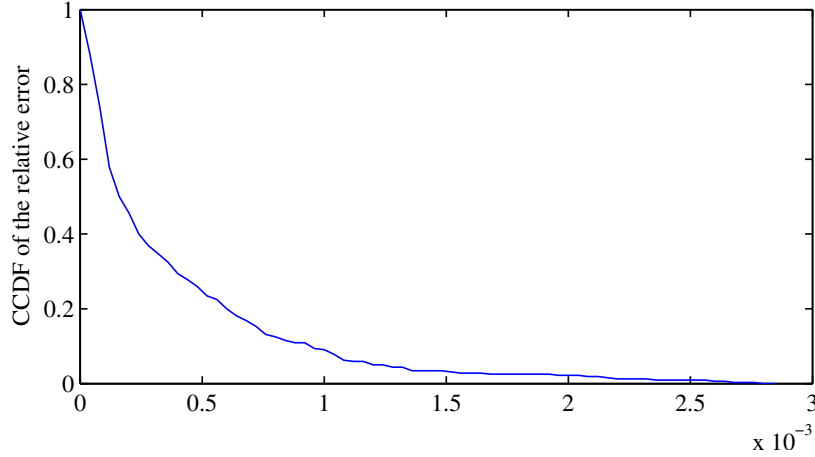


Figure 3: The CCDF of the relative error induced by the approximation (8).

## 8 Numerical Results

The models presented in Sections 5 and 6 can be seen as a generalization of those presented in [10]. As a matter of curiosity, we will compare in this section the results obtained with all these models when considering an environment that is known to violate the exponential assumption on peers on-times made in [10]. This allows us to see whether the models in [10] are robust against a violation of this assumption. Once this question addressed, we solve numerically our models to evaluate the lifetime and availability of data stored on P2PSS running in different contexts. Throughout the numerical computations, we consider both centralized- and distributed-recovery implementations.

### 8.1 Parameter values

Our mathematical models have been solved numerically using a set of parameters values.

**Network parameters**  $\lambda$ ,  $\{p_i, \mu_i\}_{i=1, \dots, n}$  and  $p$ . We consider three sets of values that represent three different environments. These correspond to three data sets that have been studied in the literature. The sets *CSIL* and *Condor* have been collected by Nurmi, Brevik and Wolski [20]. The *CSIL* set reports uptime of machines in the Computer Science Instructional Laboratory (CSIL) at the University of California, Santa Barbara. As for the *Condor* set, it reports CPU idle times of peers in a Condor pool [8] at the University of Wisconsin, in other words, it reports the availability of peers to perform an external job (the Condor pool offers processing time to the whole Internet). This can be seen as the time during which a peer may participate in a storage system. The *All-pairs-ping* set

Table 2: Data sets characteristics and corresponding peers parameters values

Data set	<i>CSIL</i>	<i>Condor</i>	<i>All-pairs-ping</i>
Context	LAN	Internet	PlanetLab
Covered period	8 weeks	6 weeks	21 months
Number of peers	83	210	200–550
On-times distribution	$H_3$ [20] (best fit)	$H_2$ [20] (best fit)	Exp. [22] (reasonable)
On-times parameters			
$p_1$	0.464	0.592	1
$p_2$	0.197	0.408	—
$p_3$	0.339	—	—
$1/\mu_1$ (hours)	250.3	0.094	181
$1/\mu_2$ (hours)	1.425	3.704	—
$1/\mu_3$ (hours)	33.39	—	—
Mean on-time (hours)	127.7	1.567	181
Mean off-time (hours)	48	0.522	61
Percentage of on-times	0.727	0.75	0.750
Persistence probability $p$	0.3	0.8	0.3

has been obtained by Stribling [26] after the processing of ping requests between each pair of PlanetLab [21] nodes. Each node pings every other node roughly 4 times an hour. A 10-probes ping is considered successful only if at least one probe response was received.

The sets *CSIL* and *Condor* are best fit by a hyper-exponential distribution according to the analysis in [20], even though they report different flavors of peer “availability”. An exponential distribution is found to “reasonably” fit the *All-pairs-ping* data set in [22]. The basic characteristics of the three data sets considered here and the corresponding values of the peers parameters are reported in Table 2. Out of the three mentioned scenarios, *Condor* experiences the highest dynamics environment. This behavior has been reported elsewhere concerning peers on the Internet. For instance, it has been observed in [4, 5] that on average peers join/leave the Internet 6.4 times per day and that sessions times are typically on the order of hundreds of minutes on average. In this report, the *Condor* system will mirror the Internet context and *CSIL* and PlanetLab environments will mirror a stable environment such as local area or research laboratory networks where machines are usually highly available. As an exponential distribution is found to “reasonably” fit the peers availability in the *All-pairs-ping* data-set, PlanetLab-like systems can be studied using the models presented in [10] while the *CSIL* and *Condor* contexts need the more general models developed in this report. Justifying this last point is the objective of the next section.

The value of  $\lambda$ , or equivalently the mean off-time, has been set to have the same *peers availability* across all environments. This measure, given in row 16 of Table 2, is the probability of finding a peer connected or equivalently the percentage of on-times in a peer life cycle. We have set  $p = 0.8$  in the *Condor* scenario as peers churn rate is very high and

$p = 0.3$  otherwise, namely in the *CSIL* and *All-pairs-ping* scenarios. This is to reflect that disconnections in stable environments are likely due to software or hardware problems.

**Protocol parameters  $s$ ,  $r$  and  $k$ .** Nowadays, block sizes in P2P *storage* systems are usually set to either 4MB, 8MB or 9MB and fragment sizes are set somewhere between 256KB and 1MB. A helpful factor to choose from these values can be the average size of the stored files in the system, so that the fragmentation overhead associated with the transmission of data is still negligible with respect to the files sizes. Concerning *CSIL*- and *Condor*-like systems, we will consider block sizes of 4 MB and fragment sizes of 1 MB if erasure codes (EC) are used (so that  $s = 4$ ) and of 1.35 MB if regenerating codes are used. In the *CSIL* scenario. In [10] where results in the PlanetLab context are reported, we considered block sizes of 8MB and fragment sizes of 1MB and then  $s = 8$ . In the *CSIL* scenario where peers churn is low, we vary the redundancy  $r$  from 1 to  $1.5s = 6$ . In the high dynamic scenario (*Condor*), we vary the redundancy  $r$  from 1 to  $3s = 12$  (resp. to  $1.5s = 6$ ) when the recovery is distributed (resp. centralized). In all the considered scenarios, we vary the threshold  $k$  from 1 to  $r$ .

Observe that the optimal amount of redundancy  $r$  comes as a trade-off between high data availability and high storage efficiency and depends on the recovery threshold  $k$ . Smaller threshold values allow for smaller amounts of redundant data at the expense of higher bandwidth utilization. The trade-off here is between efficient storage use (small  $r$ ) and efficient bandwidth use (large  $k$ ).

**Recovery process parameters  $\alpha$  and  $\beta$ .** Fragments download/upload times depend on the upload/download capacities of the peers and the central authority when needed. The measurement study [25] of P2P *file sharing* systems, namely Napster and Gnutella, shows that 78% of the users have downstream bottleneck of at least 100 Kbps. Furthermore, 50% of the users in Napster and 60% of the users in Gnutella use broadband connections (Cable, DSL, T1 or T3) having rate between 1Mbps and 3.5Mbps. Moreover, a recent experimental study [15] on P2P VoIP and file sharing systems shows that more than 90% of users have upstream capacity between 30 Kbps and 384 Kbps, where the downstream is of the order of some Mbps (like Cable/ADSL). When RC is enforced, the recovery process is faster due to the fact that the size of the downloaded fragments is smaller. Recall that with EC a new peer downloads  $s$  fragments each of size 1 MB, whereas with RC it downloads  $s$  pieces each of size  $\geq \frac{s}{s^2 - s + 1} \cdot S_B / s = 0.308$  MB. Therefore, and based on the above two mentioned studies and on the simulation results presented in [11], we assume that in EC based systems  $1/\alpha = 56$  seconds for *CSIL* context and 88 seconds for *Condor* context, and in RC based systems  $1/\alpha = 23$  seconds for *Condor* context.

## 8.2 Comparison with the models of [10]

As mentioned previously, the models presented in [10] are a special case of the models developed here, namely when the number of phases of the hyper-exponential distribution of on-times is  $n = 1$ . To avoid any ambiguity, the models presented in [10] will be referred to as “basic” models whereas the ones developed in Section 5 and 6 will be referred to as “general”

models. Because of the reduced state-space, solving the basic models is much less time consuming than solving the general models. The basic models describe well PlanetLab-like environments. However, one question remains: do they model any environment?

To answer this question, we deliberately select a scenario in which peers have been identified to have a non-exponential on-times distribution, namely the *Condor* scenario, and evaluate the lifetime of a block of data  $D$  using both models and compare the results. In [20], a 2-stage hyper-exponential distribution is found to best fit the *Condor* data set, but the authors identify as well the parameter of the exponential distribution that best fits the same data.

Table 3 reports the expected data lifetime obtained with the distributed-recovery implementation for  $s = 4$ ,  $1/\lambda = 0.522$  hour,  $1/\alpha = 22$  seconds,  $p = 0.8$  and different amounts of redundancy  $r$  and recovery thresholds  $k$ . Results provided by the general model with  $1/\mu_1 = 0.094$  hours,  $1/\mu_2 = 3.704$  hours,  $p_1 = 0.592$  and  $p_2 = 1 - p_1$  are in column 3; those given by the basic model with  $1/\mu = 1.543$  hours (best exponential fit found in [20]) and  $1/\mu = 1.567$  (first moment of the  $H_2$  distribution) can be found in columns 4 and 6 respectively. The relative error between  $E[T_d(\mathcal{E}_{s+r})]$  (general model; column 3) and  $E[T_d(s+r)]$  (basic model [10]; columns 4 and 6) are reported in columns 5 and 7.

Table 3 reveals that the basic model returns substantially different results than those of the general model. Since the distribution of peers on-times is hyper-exponential in the *Condor* scenario, the results obtained through the general model are the correct ones. We conclude that the basic model presented in [10] does not capture the essence of the system performance when peers on-times are not exponentially distributed. *Henceforth, we will use the basic model in scenarios with the All-pairs-ping characteristics, and the general model in scenarios with the characteristics of either CSIL, or Condor.*

Table 3: Expected data lifetime (expressed in hours) in a *Condor* scenario using a distributed-recovery scheme. Comparison between  $E[T(\mathcal{E}_{s+r})]$  (extended model) and  $E[T_h^e(s+r)]$  (simple model).

$s = 4$		$H_2$ fit [20] $E[T(\mathcal{E}_{s+r})]$	Exponential fit [20] $E[T_h^e(s+r)]$ error		equating 1st moments $E[T_h^e(s+r)]$ error	
$k = 1$	$r = 2$	1.437	0.78	-45.7%	1.017	-29.2%
	$r = 4$	5.866	3.453	-41.1%	4.09	-30.2%
	$r = 6$	15.751	14.04	-10.8%	14.44	-8.32%
$k = 2$	$r = 2$	0.729	0.492	-3.5%	0.633	-13.1%
	$r = 4$	3.689	2.34	-36.5%	2.74	-25.7%
	$r = 6$	12.263	10.464	-14.67%	10.732	-12.48%

Table 4: Expected lifetime and first availability metric

<b>Condor context</b> $s = 4$		$E[T(\mathcal{E}_{s+r})]$ (in days)		$M_1(\mathcal{E}_{s+r})$	
		cent. repair $1/\alpha = 22$ sec $1/\beta = 6$ sec	dist. repair $1/\alpha = 22$ sec —	cent. repair $1/\alpha = 22$ sec $1/\beta = 6$ sec	dist. repair $1/\alpha = 22$ sec —
$k = 1$	$r = 2$	0.365	5.34e-02	5.786	5.662
	$r = 4$	20.769	0.175	7.680	7.244
	$r = 5$	129.551	0.305	8.627	7.980
	$r = 6$	730.132	0.526	9.574	8.679
	$r = 10$	—	3.933	—	11.104
	$r = 12$	—	9.558	—	12.117
$k = 2$	$r = 2$	0.123	1.92e-02	5.389	4.940
	$r = 4$	6.955	9.62e-02	7.339	6.519
	$r = 5$	45.901	0.187	8.311	7.290
$k = 4$	$r = 4$	0.222	1.86e-02	6.318	5.212
<b>CSIL context</b> $s = 4$		$E[T(\mathcal{E}_{s+r})]$ (in months)		$M_1(\mathcal{E}_{s+r})$	
		distributed repair $1/\alpha = 22$ sec $1/\alpha = 56$ sec		distributed repair $1/\alpha = 22$ sec $1/\alpha = 56$ sec	
$k = 1$	$r = 2$	3.002	0.852	5.980	5.959
	$r = 4$	209.36	21.75	7.962	7.919
$k = 2$	$r = 2$	0.202	9.94e-02	5.011	5.000
	$r = 4$	24.42	4.097	6.985	6.952
$k = 4$	$r = 4$	0.162	7.878e-02	5.053	5.046

### 8.3 Performance analysis

We have solved numerically (4), (5), (7), (9) and (10) given that all  $s+r$  fragments of  $D$  are initially available, considering either *Condor* or *CSIL* context, and either the centralized or distributed recovery scheme. Results are reported partially in Table 4. Results in PlanetLab (*All-pairs-ping*) context using the basic model can be found in [10].

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, while an erasure code is enforced, yields a significantly smaller expected data lifetime than the centralized scheme for the same redundancy mechanism, especially when peers churn rate is high; cf. columns 3-4 in Table 5. Observe also how the performance deteriorates as peer churn becomes more important: compare for instance in Table 4 rows 6 vs. 19, and 12 vs. 22; these correspond to the same storage overhead  $r/s = 1$  and the same value of recovery threshold  $k = 1$  and 2 respectively, but the context is different.



Table 5: Expected lifetime in *Condor* context for centralized or distributed repair scheme and either EC or RC are enforced

<b>Condor context</b>		E[ $T(\mathcal{E}_{s+r})$ ] (in days)		
recovery		cent.	dist.	dist.
redundancy		EC	EC	RC
$1/\alpha$ in sec.		88	88	23
$1/\beta$ in sec.		6.3	—	—
$k = 1$	$r = 2$	0.365	5.34e-02	1.607
	$r = 4$	20.769	0.175	34.7
	$r = 5$	129.551	0.305	144.15
	$r = 6$	730.132	0.526	533.22
$k = 2$	$r = 2$	0.123	1.92e-02	0.43
	$r = 4$	6.955	9.62e-02	8.21
	$r = 5$	45.901	0.187	37.81
$k = 4$	$r = 4$	0.222	1.86e-02	0.33

Observe how the use of regenerating codes improves very well the performance of the system even in dynamic context; cf. columns 4-5 and 3-5. This is due to the fact that each new peer with RC downloads the size of one fragment unlike the case of EC where each peer downloads the size of the whole block of data in order to recover one fragment. However, in a stable context like *CSIL*, EC with the distributed repair scheme performs well as shown in Table 4.

We conclude that *when peers churn rate is high, only the centralized repair scheme can be efficient if erasure code is used as redundancy mechanism, as long as the storage overhead is kept reasonable small (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. Regenerating codes scheme is very promising for the storage objective in dynamic context even with distributed repair.

#### 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 distributed and the context is similar to *CSIL*. We have picked two contour lines of each of the performance metrics studied in this work and report them in Fig. 4. Consider point A (resp. B) which corresponds to  $r = 5$  and  $k = 3$  (resp.  $k = 2$ ). Recall that  $s = 4$  (for both points). Selecting point A (resp. B) as the operating point of the P2PSS ensures the following: given that each data is initiated with  $s + r = 9$  available fragments, then (i) the expected data lifetime is 22.25 (resp. 188.91) months; (ii) 23.7% (resp. 3.13%) of the stored data would be lost after six months; (iii) as long as  $D$  is not lost, 6.486 (resp. 7.871) fragments of  $D$  are expected to be available in the system; (iv) during 99.9992% (resp. 99.9999%) of its lifetime,  $D$  is available for download; and (v) during 99.79% (resp. 99.7%)

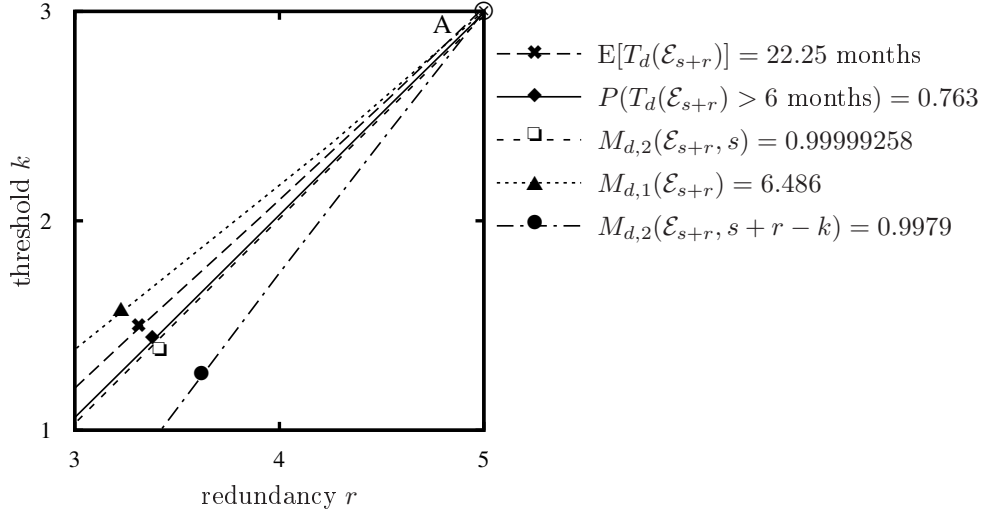
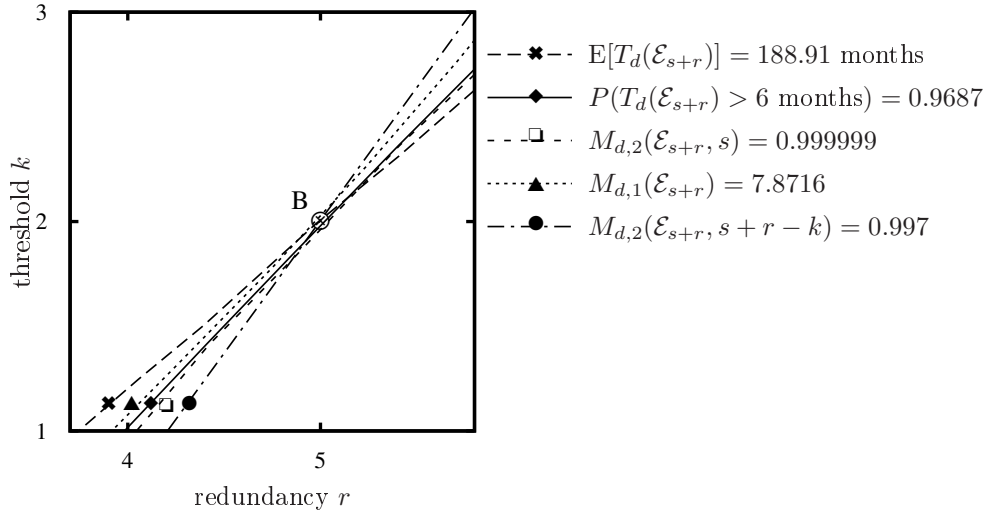

 (a) Settings of point A:  $s = 4$ ,  $r = 5$  and  $k = 3$ 

 (b) Settings of point B:  $s = 4$ ,  $r = 5$  and  $k = 2$ 

Figure 4: Contour lines of performance metrics (CSIL context, distributed repair).

of the lifetime of  $D$ , at least  $s + r - k = 6$  (resp.  $s + r - k = 7$ ) of its fragments are available. Observe that the storage overhead,  $r/s$ , is 1.25 for both operating points and it is the *lazy*

policy that is enforced ( $k > 1$ ). Observe how the performance metrics improve when  $k$  is decreased, even by one. However, this incurs more bandwidth use because the recovery will be more frequently triggered.

## 9 Conclusion

We have proposed general analytical models for evaluating the performance of two approaches for recovering lost data in distributed storage systems and three redundancy schemes. Numerical computations have been performed to illustrate several issues of the performance. We conclude that, using our theoretical framework, it is possible to tune and optimize the system parameters for fulfilling predefined requirements. 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 regardless the redundancy scheme. This is in contrast with the case of highly dynamic environments, where the distributed-repair scheme is inefficient with erasure codes 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. Regenerating codes scheme is very promising for P2PSS applications. However, the analysis of the overhead cost (e.g. computation, bandwidth and complexity cost) resulting from the different redundancy schemes with respect to their advantages (e.g. simplicity), is left for future work.

## References

- [1] S. Alouf, A. Dandoush, and P. Nain. Performance analysis of peer-to-peer storage systems. In *Proc. of 20th International Teletraffic Congress (ITC)*, volume 4516 of *LNCS*, pages 642–653, Ottawa, Canada, 17–21 June 2007.
- [2] F. Baskett, K.M. Chandy, R.R Muntz, and F.G Palacios. Open, closed, and mixed networks of queues with different classes of customers. *J. ACM*, 22(2):248–260, 1975.
- [3] R. Bhagwan, D. Moore, S. Savage, and G.M. Voelker. Replication strategies for highly available peer-to-peer storage. In *Future Directions in Distributed Computing*, volume 2584 of *Lecture Notes in Computer Science*, pages 153–158. Springer, 2003.
- [4] R. Bhagwan, S. Savage, and G. Voelker. Understanding availability. In *Proc. of 2nd IPTPS*, Berkeley, California, February 2003.
- [5] 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.

- [6] J.W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. *SIGCOMM Comput. Commun. Rev.*, 28(4):56–67, 1998.
- [7] Y. Chen, J. Edler, A. Goldberg, A. Gottlieb, S. Sobti, and P. Yianilos. A prototype implementation of archival Intermemory. In *Proc. of ACM DL '99*, pages 28–37, Berkeley, California, August 1999.
- [8] Condor: High throughput computing. <http://www.cs.wisc.edu/condor/>, 2007.
- [9] F. Dabek, F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proc. of ACM SOSP '01*, pages 202–215, Banff, Canada, October 2001.
- [10] A. Dandoush, S. Alouf, and P. Nain. Performance analysis of centralized versus distributed recovery schemes in P2P storage systems. In *Proc. of IFIP/TC6 Networking 2009*, volume 5550 of *LNCS*, pages 676–689, Aachen, Germany, May 11–15 2009.
- [11] A. Dandoush, S. Alouf, and P. Nain. Simulation analysis of download and recovery processes in P2P storage systems. In *Proc. of 21st International Teletraffic Congress (ITC)*, Paris, France, September 2009.
- [12] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. In *Proc. of 26th IEEE Conference on Computer Communications (INFOCOM)*, Anchorage, Alaska, USA, May 6–12 2007.
- [13] A.V. Goldberg and P.N. Yianilos. Towards an archival Intermemory. In *Proc. of ADL '98*, pages 147–156, Santa Barbara, California, April 1998.
- [14] C. Grinstead and J. Laurie Snell. *Introduction to Probability*. American Mathematical Society, 1997.
- [15] 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.
- [16] 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.
- [17] H. Kobayashi and B. L. Mark. On queueing networks and loss networks. In *Proc. 1994 Annual Conference on Information Sciences and Systems*, Princeton, NJ, March 1994.
- [18] J. Kubiawicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proc. of ACM ASPLOS*, pages 190–201, Boston, Massachusetts, November 2000.
- [19] M.F. Neuts. *Matrix Geometric Solutions in Stochastic Models. An Algorithmic Approach*. John Hopkins University Press, Baltimore, 1981.

- [20] D. Nurmi, J. Brevik, and R. Wolski. Modeling machine availability in enterprise and wide-area distributed computing environments. In *Proc. of Euro-Par 2005*, volume 3648 of *LNCS*, pages 432–441, Lisbon, Portugal, August 2005.
- [21] PlanetLab. An open platform for developing, deploying, and accessing planetary-scale services. <http://www.planet-lab.org/>, 2007.
- [22] S. Ramabhadran and J. Pasquale. Analysis of long-running replicated systems. In *Proc. of IEEE Infocom*, Barcelona, Spain, April 2006.
- [23] I.S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of SIAM*, 8(2):300–304, June 1960.
- [24] 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.
- [25] S. Saroiu, P.K. Gummadi, and S.D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proc. of Multimedia Computing and Networking (MMCN)*, San Jose, California, January 2002. Best Paper Award.
- [26] J. Stribling. PlanetLab - All Pairs Pings. [http://pdos.csail.mit.edu/~strib/pl\\_app](http://pdos.csail.mit.edu/~strib/pl_app), 2005.
- [27] UbiStorage. <http://http://www.ubistorage.com>.
- [28] B. Warner, Z. Wilcox-O’Hearn, and R. Kinninmont. Tahoe: A Secure Distributed Filesystem. <http://allmydata.org/~warner/pycon-tahoe.html>.
- [29] H. Weatherspoon and J. Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In *Proc. of IPTPS '02, Cambridge, Massachusetts*, volume 2429 of *Lecture Notes in Computer Science*, pages 328–337, March 2002.



---

Unité de recherche INRIA Sophia Antipolis  
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes  
4, rue Jacques Monod - 91893 ORSAY Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique  
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

---

Éditeur  
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)  
<http://www.inria.fr>  
ISSN 0249-6399