Computer Networks 59 (2014) 1–16

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

On the interplay between data redundancy and retrieval times in P2P storage systems

Lluis Pamies-Juarez [a,*,1], Marc Sanchez-Artigas [b], Pedro García-López [b], Rubén Mondéjar [b], Rahma Chaabouni [b]

[a] HGST Research. HGST, A Western Digital Company
[b] Universitat Rovira i Virgili, Department of Computer Engineering and Maths, Tarragona, Spain

ARTICLE INFO

Article history:
Received 4 November 2012
Received in revised form 25 July 2013
Accepted 6 December 2013
Available online 13 December 2013

Keywords:
P2P storage
Retrieval times
Storage codes
Distributed systems

ABSTRACT

Peer-to-peer (P2P) storage systems aggregate spare storage resources from end users to build a large collaborative online storage solution. In these systems, however, the high levels of user churn—peers failing or leaving temporarily or permanently—affect the quality of the storage service and might put data reliability on risk. Indeed, one of the main challenge of P2P storage systems has traditionally been how to guarantee that stored data can always be retrieved within some time frame. To meet this challenge, existing systems store objects with high amounts of data redundancy, rendering data availability values close to 100%, which in turn ensure optimal retrieval times (only constrained by network limits). Unfortunately, this redundancy reduces the overall net capacity of the system and increases data maintenance costs. To alleviate these problems data redundancy can be reduced at the expense of lengthening retrieval times. The problem is that both the rewards and disadvantages of doing so are not well understood. In this paper we present a novel analytical framework that allows us to model retrieval times in P2P storage systems and describe the interplay between data redundancy and retrieval times for different churn patterns. Using availability traces from real P2P applications, we show that our framework provides accurate estimation of retrieval times in realistic environments.

1. Introduction

Cloud storage solutions like Dropbox or Google Drive are a family of increasingly popular online services that allow users to store their personal data off-site, achieving better reliability than existing local storage solutions, while offering additional services such as the possibility to synchronize data across multiple devices, or to collaboratively share data between users. However, despite the numerous advantages of cloud storage services, there are still several impediments for a massive and transversal utilization of such services. For example, companies and governments see security and privacy concerns as their major barriers. The main concern of end users is, however, the cost of storing an ever-growing amount of personal data.

Peer-to-peer (P2P) storage systems, originally proposed one decade ago to aggregate users' spare storage resources into large distributed storage systems [1,9,28], have now the potential to address the cost and privacy concerns of Cloud solutions. For instance, P2P-like infrastructures, very often coupled with encryption, have already been used to address privacy concerns over Online Social Networks [3,7] and Cloud storage [12,13], amongst other works.

In P2P storage systems, users contribute their local resources to obtain free online storage capacity in return

* Corresponding author.
E-mail addresses: lpjuarez@ntu.edu.sg (L. Pamies-Juarez), marc.sanchez@urv.cat (M. Sanchez-Artigas), pedro.garcia@urv.cat (P. García-López), ruben.mondejar@urv.cat (R. Mondéjar), rahma.chaabouni@urv.cat (R. Chaabouni).

[1] This work was done while the author was a PhD candidate in Universitat Rovira i Virgili, Spain.

1389-1286/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.bjp.2013.12.005

[6,24]. In these systems, data can be spread to several nodes, with no node having access to the whole original data. However, storage resources in P2P systems are less reliable and highly prone to temporary disconnections, which require storing data with large amounts of redundancy to mask temporary failures. The simplest way to store data with redundancy is to store multiple copies (or replicas) of data in different nodes. However, more elaborated redundancy schemes based on *storage codes* can improve the reliability achieved by simple replication, minimizing both the storage and network costs associated with data redundancy [27,38,39].

Irrespective of how redundancy is introduced into the system, the key question in the design of any P2P storage system is: *How much redundancy needs to be introduced to guarantee a certain storage quality and data reliability?* Typically, existing P2P storage systems take the simplest approach; they determine the amount of redundancy required to maintain *data availability* close to 100% at all times. Loosely speaking, they introduce enough redundant data pieces to guarantee that enough pieces are always present in the set of online nodes to reconstruct the original data. As a side effect, high data availability in turn guarantees that data can be retrieved in optimal time, because there are always enough online nodes from which download data that users do not need to wait for node re-connections.[2] Here is where a thorough understanding of the interplay between data redundancy and retrieval times makes sense, a relationship that has been not well understood in the literature and our paper starts shedding some light on it. A good understanding of this relationship is crucial for future research as discussed below.

1.1. Motivation

It is well-known that guaranteeing high data availability requires high storage costs in terms of storage overheads and maintenance communication [2]. The latter occurs because data redundancy can be lost due to permanent failures, and the mechanisms required to repair it might require large amount of network traffic. This is particularly visible in the case of using *storage codes*. While the replacement of replicated data is trivially a copy, in storage codes, every bit of new data is the result of a coding operation over several other bits of data, usually stored across several nodes. This introduces additional communication costs needed to retrieve the bits to be coded, which translates into network traffic.

Due to these high costs, minimizing the amount of redundancy used is crucial to efficiently build both hybrid and pure P2P storage systems. According to [2], reducing the amount of redundancy can bring the following benefits: (1) increasing the overall storage capacity of the system and/or (2) reducing the amount of communication required for data maintenance. However, less redundancy means less data availability, which in turn yields longer object retrieval times, the subject of study of this work because of the associated negative consequences of an underestimation of the required redundancy. Concretely, longer retrieval times can affect storage systems in two different ways:

1. It can cause users to suffer poor retrieval performance.
2. Data could be destroyed faster than maintenance processes are able to repair it, which could be catastrophic.

So, when reducing redundancy, storage systems must be very cautious to not compromise these two aspects. It is crucial then, to predict the effects that redundancy have on object retrieval times. Unfortunately, to the best of our knowledge, there is no study that *explicitly* measures the interplay between the amount of redundant data and retrieval times in P2P storage systems, which constitutes the original contribution of this work. Unlike the relationship between redundancy and data availability, which is well understood by the existing literature [2,20,27], this relationship is more tricky.

To whet the reader's appetite, Fig. 1 plots a general sketch of the existing relationship between data redundancy, data availability, and retrieval times. Clearly, there exists a cross-over point beyond which decreasing redundancy can be catastrophic. However, this figure also provides an interesting insight that has never been discussed before and can be of great help to system architects: *the fact that by slightly relaxing retrieval times, one might reduce the amount of redundancy significantly without compromising data availability too much.*

To shed some light on understanding the problems of measuring the effects of data redundancy in retrieval times we depict a small example in Fig. 2. In this example, a single data object is redundantly stored using a storage code that spreads it into several storage nodes (eight nodes in the left-hand side example and six nodes in the right-hand side example). Users can retrieve the stored object by contacting and downloading the data stored in any four nodes, and each on/off process describes the availability periods of each node. In the first scenario, we have a storage system where data is stored with high data availability: there are always four nodes online, and hence, the stored data can always be retrieved in optimal time. However, in the second scenario, data is stored with low data availability,

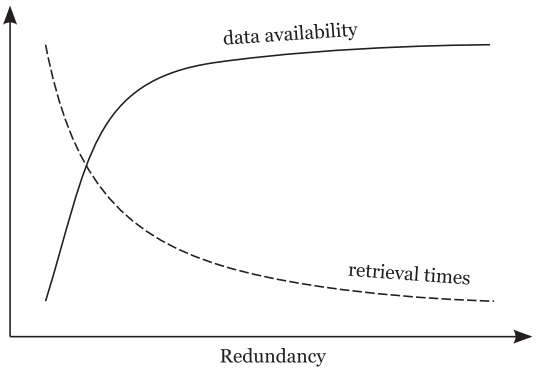


Fig. 1. Simple scheme showing the relationship between data redundancy, data availability, and retrieval times.

[2] The retrieval time is only constrained by the network bandwidth conditions.

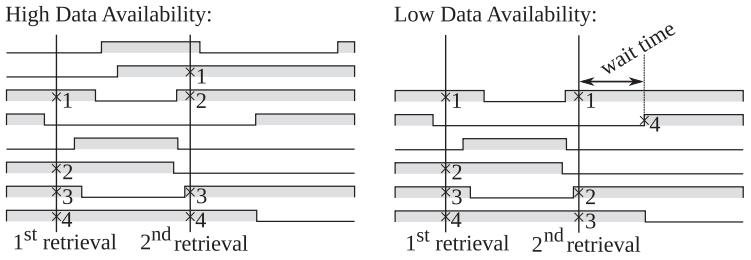


Fig. 2. Examples of two object retrievals in an encoded P2P storage system. In the first case data is spread across eight nodes, achieving a high data availability. In the second case data is spread across six storage nodes, achieving a lower data availability than in the first case. Each on/off process represents the online/offline sessions of a storage node.

and sometimes, less than four nodes are found online at the same time. In that case, the retrieval process needs to wait for one node to re-connect before being able to download the last block, which lengthens the overall retrieval time.

Although this example simplifies most of the inherent real complexities in retrieval processes, such as parallel block downloads, or network inefficiencies, it clearly reflects the tricky relationship between data redundancy (and thus data availability) and retrieval times.

1.2. Contributions

In this paper, we present a basic analytical framework that allows to model retrieval times in P2P storage systems. Due to the complexity of the stochastic processes behind retrieval processes, we make some assumptions to simplify them and obtain an estimator of the retrieval time distribution. As a second contribution, we evaluate the accuracy of our model using availability traces from real P2P systems, showing that our model offers an accurate estimator. For simplicity, but without loss of generality, we only consider pure P2P storage systems, leaving the practical implications of how to combine P2P storage systems with cloud or other hybrid systems for future research works. As a final conclusion, we discuss how our framework can help storage system designers provision their systems with the optimal data redundancy required to satisfy specific storage quality needs. Using our framework, for example, designers can find the minimum data redundancy that guarantees a targeted retrieval time performance, or in backup systems, find the minimum data redundancy that guarantees that data can be repaired faster than it is lost.

We believe that the reason why existing P2P storage systems have not considered the relationship between data redundancy, data availability and retrieval times is because they discarded the possibility to target flexible retrieval time guarantees. Instead, existing P2P storage systems base their service quality on guaranteeing the highest possible data availability, and hence, optimal repair times. However, by reducing the amount of redundant data, P2P storage systems can achieve trade-offs that allow to reduce their costs while still offering a good storage service to their users. For example, in backup applications where data is occasionally read, users can tolerate long reconstruction times as long as their data durability is not compromised. Or other storage systems can offer more storage capacity to those users accepting some loss in their data retrieval performance. We hope that this paper will give rise to further discussion on these critical aspect of P2P storage systems.

The rest of this chapter is organized as follows. In Section 2 we present the related work. Section 3 introduces the P2P storage model that we will use along the paper. Section 4 states our problem statement. Section 5 provides the analytical framework to model retrieval times, as well as our retrieval time estimator. In Section 6 we evaluate our estimator, and in Section 8 we state our conclusions.

2. Related work

The understanding of the interplay between redundancy and retrieval times is a topic that has not been explored much in the literature on P2P storage. The closest work to our proposed framework was done by Toka et al. in [34,35], where the authors presented a backup scheduling algorithm to minimize the time required to backup some amount of data in peer-assisted storage systems. Although their scheduling algorithm might be used to optimize download times, the authors did not investigate how to analyze the intricate interplay between data redundancy and retrieval times, as we do in this paper. In a more recent work, Toka et al. [33] indirectly discusses a variant of the problem discussed in this paper for P2P backup systems. This work can be seen as complementary to ours. The differences come from the very nature of backup applications, which target data durability instead of data availability, and involve the bulk transfer of large quantities of data. In this sense, our focus is on understanding to what extent smaller redundancy factors affect data availability and retrieval times for data objects of moderate size, an issue that has not been addressed before in the literature.

Besides that, and significantly far from P2P storage, there are some studies that examine retrieval times in BitTorrent [11,19,25], which is a P2P file distribution system rather than a storage system. The main similarity between BitTorrent and a typical P2P storage system based on storage codes is that files are broken into equally sized chunks which are downloaded from multiple peers, who in turn can be either online or disconnected as any conventional P2P application. However, P2P storage systems, and

particularly those based on storage codes, present some singularities that require a different analytical treatment which invalidate the results from BitTorrent studies. On the one hand, BitTorrent studies investigate retrieval times in the short term. This implies that peer disconnections are seen as permanent failures. The direct consequence of this is that these studies ignore the option that data can be reintegrated when a peer re-joins the system. In P2P storage systems, however, what is interesting is the long term behavior of the system. For this reason, transient disconnections must be considered as well because when a disconnected peer re-joins the system, it could make available one of the missing chunks or blocks required to reconstruct the original file.

On the other hand, in BitTorrent multiples nodes can have a full replica of the file, the nodes known as *seeders* in the BitTorrent jargon, which substantially differs compared to what occurs in P2P storage systems. Typically, in these systems, storage codes are used to disperse data among several nodes, with no node having a complete replica of the file. Due to the lack of seeders, retrieval processes must mandatorily contact several nodes in order to reconstruct the original file, which is not necessarily true in BitTorrent.

Once explained the main drawbacks of attempting to extrapolate the results of prior works in BitTorrent to predict retrieval times in storage systems, we are ready to comment on the most important works in that area. Ramachandran and Sikdar [25] studied the times required to retrieve replicated objects in BitTorrent using queuing theory. The result of their analysis was a framework able to evaluate and predict transfer times along with data query times. Non-surprisingly, and because their focus was not on P2P storage, they did not consider the impact of low data availabilities and the effect of storage codes on redundancy. Gaeta et al. [11] proposed a stochastic fluid model to analyze file download times. Their analysis was focused on the impact of four different parameters: file popularity, peer selection policies, available bandwidths and concurrent downloads. Similarly, Liao et al. [19] analyzed the same four aspects in addition to file retrieval times, but considering heterogeneity in bandwidth.

Regarding P2P storage, what has received considerable research effort has been the study of the existing tradeoff between data availability and redundancy [2,20,27]. What kind of redundancy scheme to use has been extensively discussed in the literature and many papers focused on the comparison between replication and storage codes [20,27,39]. Compared to replication, the authors of these studies concluded that erasure codes are more space efficient, but also require more maintenance traffic to reconstruct a lost block of data [20,27]. Later on, although more elaborated types of storage codes like *Regenerating Codes* [8] have been increasingly appearing, exploring the interplay between redundancy and retrieval times has remained an open question.

Finally, we want to highlight that some preliminary results of this paper were presented in [23], where we provided an estimator for the *average* retrieval time. In this paper, we refine our model and derive an estimator for the *whole* retrieval time distribution, which is easier to manage and allows us to study questions that cannot be answered with an estimate of the average retrieval time such as what is the risk of exceeding any given threshold value on the retrieval time, among other issues.

3. P2P storage model

In this section we provide the node churn model and the data redundancy model that we will use in the rest of the paper.

3.1. Application scenario

As usual, we consider a traditional P2P storage system where free storage of individual users is aggregated in order to realize persistent and highly available data storage. We assume that data owners specify a local folder with the files to store remotely. For simplicity, we assume the existence of a service that is in charge of membership management of the P2P system. While decentralized membership management and system monitoring is very appealing, it is routine practice (e.g., Wuala, CrashPlan, TotalRecall [17] and subsequent works [5,21], ...) to rely on a centralized service and a simple heartbeat mechanism to keep track of the users subscribed to the application and their availability patterns.

During a storage operation, data owners query this service to obtain remote peers that can be *potentially* used to store their data. We say "potentially" because the chosen remote peers may exhibit temporary and recurrent, periods of unavailability. Particularly, individual peers may experience *transient* and *permanent* failures and even abandon the P2P application. For this reason, the data owner periodically probes each peer storing part of its data for availability. During a read or write operation, it is the data owner himself who downloads and uploads the targeted data by *directly* contacting a sufficient number of remote peers. Consequently, data flows do not traverse any intermediate hop, which minimizes retrieval times.

Transient failures are typically caused by provider errors, power outages, or user disconnections. During transient failures, stored data is not lost, becoming only temporarily unavailable, and being reintegrated back into the system when the remote peer reconnects. Permanent failures, however, are complete node failures after which the stored data becomes unrecoverable. Even if a permanent failed node can fix the problem and rejoin the system, the stored data is never reintegrated back into the system. Such an autonomous and whimsical behavior of individual peers is what is known as "churn" in the literature.

Technically, during its *lifetime* in the system, we model the behavior of each storage peer as an alternating process between two states, namely *online* and *offline* states. From the online state, a storage peer can move to the *dead* state, which indicates a permanent failure or the abandon of the P2P system. If a dead peer can manage to fix its problem and rejoin the system, it will contain no data and will be modeled as a new joining node. Fig. 3 shows the transitions between states. Before absorption to the *dead* state, the simplicity of this model allows us to describe the behavior

Fig. 3. Transitions between different storage node states.

of the each storage peer i at time t by the renewal process $X^i = \{X_t^i\}$:

$$X_t^i = \begin{cases} 1 & \text{node } i \text{ is online at time } t; \\ 0 & \text{otherwise.} \end{cases}$$

We also assume that peers behave independently of one another and that processes X^i and X^j for any $i \neq j$ are independent. This means that users do not synchronize their arrival or departure and do not exhibit regularities such as diurnal and weekly patterns. Since the centralized service monitors the availability patterns of users, we assume that it returns a set of peers exhibiting uncorrelated lifetime characteristics when asked by data owners. This can be simply achieved by finding anti-correlated peers as shown by Kermarrec et al. in [16].

Under this model, we assume that for each process X^i, online durations or sojourn times in the *online* state S_1^i have some joint distribution $F^i(x)$ and that its offline durations or sojourn times in the *offline* state S_0^i have another joint distribution $G^i(x)$. Each of these distributions corresponds to the amount of time that node i spends at offline and online states, respectively. From Smith's theorem, we can easily obtain that the asymptotic availability a_i of each user i, i.e., the probability that it is in the system at a random instance t, is [40]:

$$a_i = \lim_{t\to\infty} \Pr\left[X_t^i = 1\right] = \frac{\mathrm{E}\left[S_1^i\right]}{\mathrm{E}\left[S_0^i\right] + \mathrm{E}\left[S_1^i\right]}, \tag{1}$$

where $\mathrm{E}\left[S_0^i\right]$ and $\mathrm{E}\left[S_1^i\right]$ represent the mean offline and online session durations, respectively. We assume that after monitoring each node for a long period of time, we can obtain a good estimate of the mean node availability, a_i.

Finally, for each process X^i, we need to characterize the *residual offtime* whose duration is described by random variable J_0^i. Loosely speaking, J_0^i represents, at any given time t, the length of the remaining time that peer i will be offline. Note that J_0^i is very important since it will determine how long a data owner v will have to wait for i to come back online after a storage operation is triggered by v. In the equilibrium, i.e., the system has evolved for sufficiently long before i joined, the residual offtime distribution of storage peer i is given by [40]:

$$\Pr\left[J_0^i < t\right] = \frac{1}{\mathrm{E}[S_0]} \int_0^t \left(1 - \Pr\left[S_0^i < u\right]\right) du. \tag{2}$$

Similarly, the *residual ontime* distribution is given by:

$$\Pr\left[J_1^i < t\right] = \frac{1}{\mathrm{E}[S_1]} \int_0^t \left(1 - \Pr\left[S_1^i < u\right]\right) du. \tag{3}$$

To simplify even further our analysis, we will make use of the result of Yao et al. [40] that proved that while each peer i has specific offline and online session distributions, both the online and offline session lengths of any randomly chosen user in the system can be described by only two general distributions. This allows to characterize the dynamic behavior of the users subscribed to the application by only two random variables S_0 and S_1. Similarly, this will allow us to replace the residuals of all peers $\{J_0^i\}$ and $\{J_1^i\}$ with random variables J_0 and J_1, respectively.

As we will see, this result reduces importantly the complexity of our analysis while still considering heterogeneous online availabilities. In Section 6, we will assume that S_0 and S_1 are Weibull variates and we will show how the aggregation of heterogeneous behaviors fits the distribution of real availability traces.

3.2. Data redundancy

To guarantee a certain storage reliability, P2P storage systems need to store data with some redundancy. It means that before storing each data object, these objects are split into blocks which are then redundantly encoded and dispersed to different storage nodes. For example, one simple solution for that is to store multiple replicas of the same data to different storage nodes. However, storage coding techniques can present higher communication and storage savings than simple replication [27].

Given a data object **o** of size m, a storage code allows to encode and store **o** into a set of n storage nodes $N_1, \ldots, N_n$, assigning an amount α of information to each node, $\alpha \leqslant m$. The data redundancy ratio (or the stretch factor) is the ratio between the amount of storage required to store one object and its original size, and it is measured as the ratio $n\alpha/m$. Once the data has been successfully stored, the storage code must provide the following two properties:

- **Data reconstruction**: The original data object can be recovered by downloading a total amount of information m from a subset of k storage nodes. It constrains the amount of data stored per node to $\alpha \geqslant m/k$. We will refer to k as the *reconstruction degree*.
- **Data repairability**: The amount of information α stored in each node can be repaired by downloading an amount β per node, from d different storage nodes, where $\beta \leqslant \alpha$. We will refer to d as the *repair degree*.

Fig. 4 depicts the basic operations of an **object retrieval** and **block repair**. The labels at the edges indicate the amount of data transmitted between nodes during each of these operations.

It is important to note that the storage codes presented here generalize the concept of data replication and that of traditional erasure codes. Replication can be modeled as a storage code where $k = 1$ and $\alpha = \beta$. Similarly, traditional erasure codes like Reed–Solomon [26] codes and its variants do not distinguish between data reconstructions and data repairs, and hence are particular instances of storage codes where $k = d$ and $\alpha = \beta$. Repairing lost blocks in both cases requires reconstructing the original data and recoding the missing parts, which entails transmitting over the network an amount of data equal to the size of the original object, m. However, more elaborated instances of storage

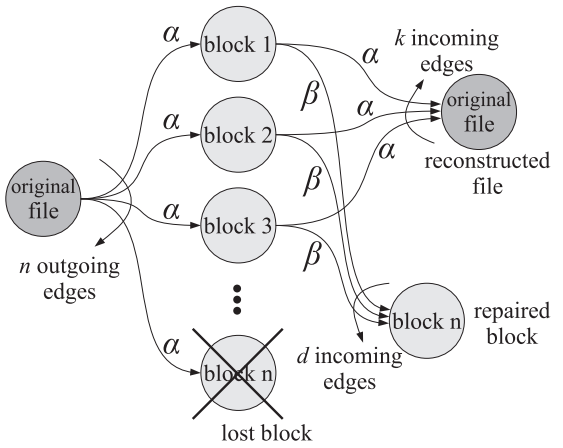


Fig. 4. Scheme for the repair and retrieve operations of Regenerating Codes.

codes like Regenerating Codes [8] can significantly reduce the amount of data transmitted during repairs. To do that Regenerating Codes allow to reduce the size of β, for larger values of d, $d \geqslant k$. In particular, Dimakis et al. [8] gave the conditions that the set of parameters $(n, k, d, \alpha, \gamma = d\beta)$ must satisfy to construct a valid Regenerating Code. Basically, once the subset of parameters: (n, k, d) is fixed, the relationship between the values of α and γ presents a trade-off curve: the larger α, the smaller γ, and vice versa. This means that it is impossible to simultaneously minimize both, communication costs and storage costs. Other recent storage codes [22] have also shown that it is possible to design storage codes with smaller repair degrees $d < k$, at the expense of increasing the amount of data stored per node α or reducing the number of k-subsets that allow to reconstruct the original data.

4. Problem statement: retrieval times as a reliability metric

Traditionally, P2P storage systems have measured the reliability of their storage service in terms of data availability and data durability, which can be defined as:

Definition 1 (*Data Availability*). Data availability is the probability of finding online k out of the n redundant blocks required to reconstruct a stored data object.

Definition 2 (*Data Durability*). Data durability is the probability of not losing a data object after being stored for some time t.

In general, the main effort of existing P2P storage systems is on maximizing data availability, guaranteeing values close to one. The reason for achieving high data availability is that it guarantees that objects can be retrieved without having to wait for node reconnections with high probability. High data availability also ensures that the redundancy lost can be repaired when nodes fail, which in turn assures long data durability. However, nodes in P2P storage systems usually have online sessions of a few hours per day. Achieving high data availability in these unstable environments might become technically impossible [20]. Or even when it is possible, it requires large amounts of redundancy, which significantly increase the costs of the storage system.

Due to the high cost of guaranteeing high data availability, the question that arises is: *Can we design P2P storage systems with low redundancy ratios while guaranteeing a certain storage quality and data reliability?*

Decreasing the amount of redundancy means reducing data availability, which in turn implies that object retrieval times will be necessarily longer. Longer retrieval times might have two undesirable effects as discussed in the introduction of this paper. On the one hand, it can cause users to experience poor retrieval performance, i.e., suffering long delays to completely download the minimum number of blocks required to reconstruct the stored data objects. On the other hand, redundancy could be destroyed faster than maintenance processes are able to repair it, which could jeopardize data durability and be catastrophic.

To ensure that the reduction on the amount of redundancy does not compromise the quality of the storage system, it is critical to predict to what extent a reduction in data redundancy turns into an increase of object retrieval times. Equipped with accurate estimates, a system architect could reduce redundancy to the exact amount that assures that:

1. Users can access the stored data within reasonable times.
2. Repairs can finish fast enough to guarantee long data durability.

In practice, the problem is that there is neither a good understanding of the relationship between redundancy and retrieval times nor an estimate to anticipate how a given amount of redundancy determines retrieval times. This issue is the target of this paper. However, before going any further, it is central to define what we mean by "retrieval time", which we do in terms of the retrieval time distribution defined as follows:

Definition 3 (*Retrieval Time Distribution*). The retrieval time distribution, $T(\ell, \varphi, n)$, is the random variate describing the time required to download ℓ blocks of size φ bytes out of a total n storage blocks.

By modeling the retrieval time distribution, we will be able to simultaneously predict **reconstruction times** and **repair times**, which are the times required to retrieve k and d blocks, respectively. In particular, we will be able to model the following two random variables:

- **Reconstruction time distribution**: Time that the reconstruction process of a storage code needs to retrieve the k blocks required to reconstruct the original stored object. We can model the reconstruction time distribution as $T(k, \alpha, n)$.
- **Repair time distribution**: Time that the repair process of a storage code needs to retrieve the d blocks required to repair a stored block. We can model the repair time distribution as $T(d, \beta, n)$.

In what follows, we will approximate the retrieval time distribution $T(\ell, \varphi, n)$ to allow P2P storage systems to reduce redundancy while not compromising neither storage quality nor data reliability.

4.1. Bandwidth model and optimal retrieval time

Before presenting the retrieval time estimator we want to define the bandwidth model that we will consider in the rest of this paper. We will use the notation $\omega\uparrow$ and $\omega\downarrow$ to respectively denote the upload and download bandwidth of each storage node. For the sake of simplicity, we will assume that all nodes have the same bandwidth capacities. By making this assumption, peer selection becomes arbitrary, and hence the order in which peers are contacted during an storage operation is irrelevant. When nodes are heterogeneous, scheduling choices are significant, since an inefficient scheduling policy can lengthen retrieval times. By only taking into account download and upload speeds, the spectrum of choices is so wide to be described mathematically. To wit, one could immediately ask what is better, to prioritize the slowest peers, based on the idea that a node with slower download speed will complete receiving its corresponding part of the data in longer time, or to prefer the fastest peers, at the risk that the slowest nodes cannot complete their data transfers when needed. In this sense, an algorithmic approach would be more appropriate [35], for we leave this question for future research.

Also, we do consider nodes with asymmetric bandwidth capacities, $\omega\uparrow \leqslant \omega\downarrow$, and the possibility to allow retrieval processes to download up to p data blocks in parallel. In that case, the maximum number of parallel downloads has been chosen to guarantee that all processes can download at maximum speed without having to share the bandwidth between them. To satisfy this condition the number of parallel process should be constrained between:

$$\left\lfloor \frac{\omega\downarrow}{\omega\uparrow} \right\rfloor \geqslant p > 0.$$

Additionally, since $\omega\uparrow \leqslant \omega\downarrow$, we assume that the retrieval time distribution $T(\ell, \varphi, n)$ will be either constrained by the upload bandwidth, $\omega\uparrow$, or by the online/offline behavior of nodes. This means that we neglect the network congestion effects or other network inefficiencies. Under these assumptions, the minimum time required to download ℓ blocks, $\tau(\ell)$, and hence, *the minimum achievable retrieval time*, is clearly given by:

$$\tau(\ell) = \frac{\varphi}{\omega\downarrow} \times \left\lceil \frac{\ell}{p} \right\rceil. \tag{4}$$

Since $\tau(\ell)$ represents the minimum time to download the required ℓ blocks without interruptions, by definition, it follows that the retrieval time distribution, $T(\ell, \varphi, n)$, must satisfy the following condition: $\Pr[T(\ell, \varphi, n) < \tau(\ell)] = 0$, i.e., no user can download faster than $\tau(\ell)$. Now we are ready to comment on the stochastic model used to approximate $T(\ell, \varphi, n)$.

5. Modeling retrieval times

We first define in Section 5.1 the stochastic model describing retrieval times in P2P storage systems. However, due to the complexity of solving this stochastic model in its generic form, we propose in Section 5.2 a simplification of the stochastic model based on assumptions of how storage nodes behave in realistic P2P scenarios. This simplification allows us to obtain in Section 5.3 an estimator of the retrieval time distribution.

5.1. A stochastic retrieval process

To stochastically compute retrieval times, it is necessary to account for the number of blocks downloaded by the retrieval process. Let $Z = \{Z_t\}_{t\geqslant 0}$ denote the number of downloaded blocks in time interval $[0, t]$. Then, it is easy to see that stochastic process Z is a pure-birth process whose first hitting time distribution to the absorbing state (which corresponds to the download of the ℓ-th block) is the retrieval time distribution $T(\ell, \varphi, n)$. That is,

$$T(\ell, \varphi, n) = \inf\{t \geqslant 0 : Z_t = \ell | Z_0 = 0\}. \tag{5}$$

Unfortunately, the computation of the first hitting time distribution of process $Z = \{Z_t\}_{t\geqslant 0}$ depends on the evolution of two other stochastic processes, which makes it hard to give a closed-form expression for (5) if no extra assumptions are made. Concretely, a general specification of process $Z = \{Z_t\}_{t\geqslant 0}$ would require describing the evolution of two other values:

- The number of online nodes out of the total n nodes storing blocks at time t, namely $X = \{X_t\}_{t\geqslant 0}$; and
- From the blocks available at time t, the number of online blocks that have not yet been downloaded at time t, namely $Y = \{Y_t\}_{t\geqslant 0}$, which corresponds to a birth–death process with state space $\{n, n-1, \ldots, 0\}$.

The dependencies between these three stochastic processes make the characterization of process $Z = \{Z_t\}_{t\geqslant 0}$ extremely complex, since the value of Z_t depends on the state of process $Y = \{Y_t\}_{t\geqslant 0}$, which in turn depends on the state of process $X = \{X_t\}_{t\geqslant 0}$, which can also become more complex if the retrieval process can download different blocks in parallel from different nodes, among other issues. Also, the description of the future behavior of the retrieval process (at least in a probabilistic sense) depends on the knowledge of the *past history* of downloads, thereby making inadequate a standard Markovian treatment of the problem. The reason is that the transition from state i to state $i+1$ in process Z, where $i \in \{0, 1, \ldots, \ell-1\}$ is the number of downloaded blocks, depends on the fact that from the set of available nodes at time t, there exits at least one block that has not been previously downloaded (which can be seen as a variant of the *Coupon collector's problem* where the coupons are the blocks to be downloaded but with the added complexity that the availability of each block varies over time).

Fig. 5 illustrates an example of the evolution of the three stochastic processes X, Y and Z to show the existing

dependencies between them. In the horizontal axis, we plot the three different types of events that make the system evolve: (S) a block is downloaded successfully, (C) an offline node connects, and (D) and online node disconnects. Every time that a block is successfully downloaded (S), the number of downloaded blocks increases while the number of non-downloaded blocks decreases. Using this example, some of the existing dependencies can be clearly identified. For instance, two consecutive node disconnections occur at times $t = 4$ and $t = 5$, respectively. Obviously, both disconnections decrease the number of online blocks, X_t, but have different implications on the value of Y_t depending upon whether the block stored in each disconnecting node has already been downloaded or not. Observe that while the disconnection at $t = 4$ decreases the number of non-downloaded blocks, the node disconnection at $t = 5$ does not reduce the number of non-downloaded blocks because that block was previously downloaded by the retrieval process, so Y_t is not decreased at $t = 5$. Finally, we can see that in time period $[6, 7)$ there are no missing blocks to download because the two online blocks have already been downloaded, so $Y_t = 0$ while $X_t = 2$. During these periods the retrieval process is waiting for node re-connections, and as we shall see in short, *node re-connections are by far the major contributors to the abrupt increase in retrieval times*.

Based on this insight, we will be able to obtain a practical yet accurate estimate of the retrieval time distribution, $\widehat{T}(\ell, \varphi, n)$, getting rid of the tremendous complexity of a general solution. In particular, we will obtain a closed-form expression to approximate the distribution of $\widehat{T}(\ell, \varphi, n)$ with good accuracy and based on reasonable assumptions.

5.2. Simplifying node churn

Several studies have analyzed the duration of online and offline sessions in P2P file sharing systems [29,31,15]. In these studies we can observe how nodes tend to have long online sessions, usually of the order of some hours. These studies give us the insight that even in these highly unstable distributed infrastructures, some nodes present long online sessions. Although these empirical studies where not targeted to P2P storage applications, we can expect that nodes in P2P storage systems have even longer online sessions because they have incentives to remain connected for long periods of time [14,24].

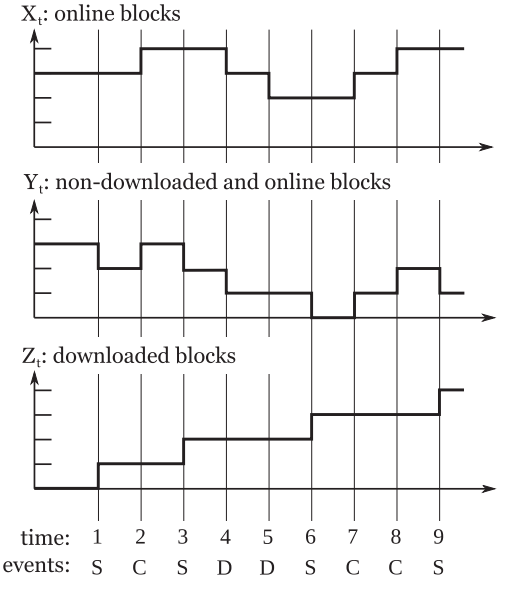


Fig. 5. Evolution of the three stochastic process used to model object retrieval times: X_t, Y_t, Z_t.

Due to these considerations, it is reasonable to expect that P2P storage systems exhibit data retrieval times shorter than online node session durations: $T(\ell, \varphi, n) \ll \mathrm{E}[S_1]$. For example, assuming an average bandwidth of 20 KBps, a user can retrieve a 60 MB object in approximately 50 min. However, typical session durations are of the order of hours, and this gap can become even larger in better-provisioned storage infrastructures. Consequently, by considering the significant difference between session durations and retrieval times, one can make the following assumption:

Assumption 1 (*Nodes change their state once*). During a retrieval process, nodes change their online state once. It means that initially online nodes will tend to disconnect, but once disconnected, they will not connect again. Similarly, initially offline nodes will tend to connect, but once connected, they will not disconnect again.

In Section 5.1, we defined the process X_t as the number of online blocks at time t. Assuming that the retrieval process starts at $t = 0$, we can use Assumption 1 to classify the number of online blocks at time t, X_t, in two different categories:

$$X_t = X_t^{\mathrm{on}} + X_t^{\mathrm{off}}.$$

On the one hand, X_t^{on} represents the online blocks stored in nodes that were online when the retrieval process started. On the other hand, X_t^{off} represents the online blocks stored in nodes that were offline when the retrieval process started. To better understand this, let us assume a simple scenario with 100 redundant blocks ($n = 100$) and with 30 online blocks when the retrieval process started. In this scenario, we initially have that $X_0^{\mathrm{on}} = 30$ and $X_0^{\mathrm{off}} = 0$; so the number of initially offline nodes can be represented as $n - X_0^{\mathrm{on}} = 100 - 30 = 70$. Since under Assumption 1 nodes can only change their state once, X_t^{on} will evolve from 30 to 0 as time tends to infinity, $\lim_{t\to\infty} X_t^{\mathrm{on}} = 0$. Analogously, X_t^{off} will evolve from 0 to 70 as time tends to infinity, $\lim_{t\to\infty} X_t^{\mathrm{off}} = n - X_0^{\mathrm{on}}$.

5.2.1. Impact of Assumption 1

As any assumption could be doubtless controversial and unrealistic, it is important to quantify to what extent the assumption that each node only changes its state once during the retrieval process can be considered unharmful to the accuracy of our estimator. To this aim, we evaluate the validity of Assumption 1. This is done by analyzing the evolution of the number of online nodes in each category, X_t^{on} and X_t^{off}, in two simulated scenarios.

In the first scenario, nodes act freely, connecting and disconnecting according to the non-simplified churn model

described in Section 3. In the second scenario, we restrict node behaviors to Assumption 1, that is, nodes connect or disconnect once during the retrieval process. In both simulations, we study the evolution of $n = 100$ nodes that have an online availability of 30%, $a = 0.3$. For both simulations, we used simple exponential session durations with the following rates: $S_1 \sim \text{Exponential}(\lambda = 0.14)$ and $S_0 \sim \text{Exponential}(\lambda = 0.06)$; which give us the following expected session durations: $\text{E}[S_1] \simeq 7.14$ hours and $\text{E}[S_0] \simeq 16.7$ hours. As in the previous example, we set the number of initially online nodes to 30 in both cases, hence, $X_0^{\text{on}} = 30$. This initial number of online blocks corresponds to the expected number of online nodes in the system.

In Fig. 6a, we depict the evolution of X_t^{on} and X_t^{off}. We use lines to describe the evolution of the nodes modeled by Assumption 1 and points for non-constrained nodes. We can see how after some hours, the number of online nodes in each category clearly differs in both scenarios. However, if we take a look at Fig. 6b where only the first 4 h of simulation are shown, we can see that the effects of Assumption 1 are almost inappreciable for the first 3 h of simulation. Since we expect retrieval times much more shorter than online sessions, $T(\ell, \varphi, n) \ll S_1$, simplifying the behavior of nodes using Assumption 1 will have small impact in the measured retrieval times while being of great help for us.

5.3. Obtaining an estimator of the retrieval time distribution

In this section, we provide an approximation of the distribution $T(\ell, \varphi, n)$, namely $\widehat{T}(\ell, \varphi, n)$ under Assumption 1 (i.e., nodes change their state once). One of the implications of this assumption is that the time needed to download a block, $\tau(1)$, as defined in (4), is required to be several orders of magnitude smaller than the average session durations $\tau(1) \ll \text{E}[S_0]$ and $\tau(1) \ll \text{E}[S_1]$. In this case it is also reasonable to make the following extra assumption:

Assumption 2 (*No transfer is canceled*). Once a block transfer starts, it is always finished. No block transfer is canceled because of a node disconnection. The storage system can choose a block size small enough to guarantee that no block transfer is ever canceled.

Given Assumptions 1 and 2 one can expect that:

1. The retrieval process can download the blocks from all nodes that were initially online when the retrieval started. Assuming that the number of initial online blocks were x, the retrieval process will take $\tau(x)$ seconds to obtain all these blocks.
2. Once an initially offline node reconnects, the data owner will be able to initially start downloading its stored block. The download of this block will take $\tau(1)$ seconds.

In addition to these two simplifying observations, we will make use of the following theorem in order to obtain our retrieval time estimator:

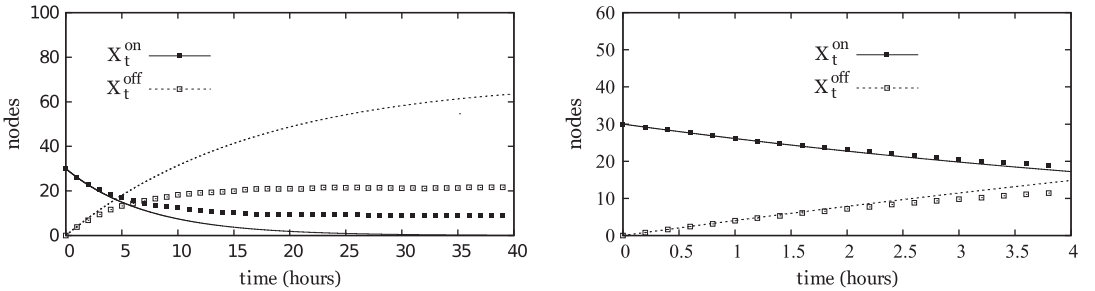
Theorem 1. *Let o be the number of offline nodes at any time t. The time elapsed until r of these o nodes reconnect, $W_{r:o}$, is distributed as:*

$$\Pr[W_{r:o} \leqslant t] = I_{\beta}(\Pr[J_0 \leqslant t]; r, o - r + 1),$$

where I_β is the regularized beta function, and $\Pr[J_0 \leqslant t]$ is the residual offline time distribution given in Eq. (3).

Proof. Let $(x_1, x_2, \ldots, x_o)$, where $x_1 \leqslant x_2 \leqslant \ldots \leqslant x_o$, be an i.i.d. sample drawn from $\Pr[J_0 < t]$. This sample represents the residual offline times of the o offline nodes. Since the sample is ordered, $W_{r:o}$ corresponds to the rth order statistic of a sample of size o. As shown in [4], $W_{r:o}$ is distributed accordingly to a regularized beta function. □

Now, lets consider a retrieval process that starts at time $t = 0$, and let X_0 be the number of online nodes at time $t = 0$. From Assumption 1 we can state that if the number of initially online nodes is $X_0 \geqslant \ell$, then the retrieval process finishes in optimal time, which is $\tau(\ell)$ seconds. Contrarily, when $X_0 < \ell$, the retrieval process takes exactly $\tau(X_0)$ seconds to download the X_0 initially online blocks. However, to download the remaining $\ell - X_0$ blocks the retrieval process needs to wait for $\ell - X_0$ nodes to reconnect, and download their blocks. In practice, the blocks are downloaded in the order in which they are coming back online, and hence, order statistics $W_{r:o}$ approximate well the delay experienced by data owners. This can be performed by the decentralized management system, which can monitor the $n - X_0$ nodes and notify the retrieval



(a) First 40 hours of simulation. (b) First 4 hours of simulation.

Fig. 6. Evolution of the number of nodes online in X^{on} and X^{off} categories. With lines we depict a simulation scenario where nodes are constrained to Assumption 1. With points we depict a simulation scenario with non-constrained nodes.

process as soon as any of the nodes become available again.

Then, it is clear that the retrieval time distribution $\widehat{T}(\ell,\varphi,n)$ depends on the initial number of online nodes, X_0. As a consequence of this, the CDF of $\widehat{T}(\ell,\varphi,n)$ can be expressed as the sum of the conditioned retrieval times:

$$\Pr\left[\widehat{T}(\ell,\varphi,n)\leqslant t\right]=\sum_{i=0}^{n}\Pr[X_0=i]\cdot\Pr\left[\widehat{T}(\ell,\varphi,n)\leqslant t\middle|X_0=i\right],$$

where $\Pr[X_0=i]$ is the probability to find i online nodes out of the total n storage nodes, which is *binomially distributed* with mean $n\cdot a$:

$$\Pr[X_0=i]=\binom{n}{i}a^i(1-a)^{(n-i)},$$

where a is the average node online availability.

To define $\Pr\left[\widehat{T}(\ell,\varphi,n)\leqslant t\middle|X_0=i\right]$, we must consider the following different cases:

$$\Pr\left[\widehat{T}(\ell,\varphi,n)\leqslant t|X_0=i\right]=\begin{cases}0 & \text{if } t<\tau(\ell),\\ 1 & \text{if } t=\tau(\ell) \text{ and } i\geqslant\ell,\\ \vartheta(i,\ell,n,t) & \text{if } t>\tau(\ell) \text{ and } i<\ell,\\ 0 & \text{otherwise.}\end{cases}$$

The first case happens with probability 0 due to the impossibility of retrieving ℓ blocks with less than $\tau(\ell)$ seconds. However, when t is equal or larger than $\tau(\ell)$, we need to consider two cases depending upon whether there are all the necessary blocks initially online ($i\geqslant\ell$) or not ($i<\ell$). When $i\geqslant\ell$, it follows by Assumption 1 and the fact that $\tau(\ell)\ll\mathrm{E}[S_1]$ (i.e., the time for downloading ℓ blocks is several orders of magnitude shorter than the online residual time) that the probability to retrieve ℓ initially online blocks with $\tau(\ell)$ seconds is always one. Finally, when $i<\ell$, the retrieval process will download the i initially online blocks with $\tau(i)$ seconds, but it will have to wait for $\ell-i$ nodes to reconnect.

Once again, observe that we assume that the time waiting for any of the $n-i$ storage nodes to reconnect will be several orders of magnitude longer than $\tau(i)$, i.e., $\tau(i)\ll\mathrm{E}[S_1]$. From Theorem 1, we know that the retrieval process will last $W_{\ell-i:n-i}$ additional seconds, plus the $\tau(1)$ seconds required to download the block from the $\ell-i$th reconnected node. We define the probability of this total time being shorter than t as $\vartheta(i,\ell,n,t)$:

$$\begin{aligned}\vartheta(i,\ell,n,t)&=\Pr[W_{\ell-i:n-i}\leqslant t-\tau(1)|W_{\ell-i:n-i}\geqslant\tau(\ell)]\\&=\frac{\Pr[W_{\ell-i:n-i}\leqslant t-\tau(1)]-\Pr[W_{\ell-i:n-i}\leqslant\tau(\ell)]}{1-\Pr[W_{\ell-i:n-i}\leqslant\tau(\ell)]}.\end{aligned}$$

In the rest of cases, the retrieval probability is zero, because otherwise it would imply breaking Assumption 1 in a way or another. Finally, putting all pieces together, we get our estimator:

$$\Pr\left[\widehat{T}(\ell,\varphi,n)\leqslant t\right]=\begin{cases}0 & \text{if } t<\tau(\ell)\\ \sum_{i=\ell}^{n}\Pr[X_0=i] & \text{if } t=\tau(\ell)\\ \sum_{i=0}^{\ell-1}\Pr[X_0=i]\vartheta(i,\ell,n,t) & \text{if } t>\tau(\ell)\end{cases}\tag{6}$$

It is important to note here that to use such an estimator, two types of information should be provided as input. On the one hand, the estimator requires to know the average node availability a to compute $\Pr[X_0=i]$. This information is empirically obtained by the centralized service. On the other hand, the estimator requires to know the shape of the residual offline time distribution $\Pr[J_0\leqslant t]$ to compute $\vartheta(i,\ell,n,t)$, since residual offtimes are involved in the computation of order statistics (see Theorem 1). Again, this information can be empirically inferred using different techniques [30,37]. Equipped with this information, it is easy first to compute $\tau(\ell)$ using Eq. (4) and then approximate the retrieval time distribution based on that value and t. That is, if $t<\tau(\ell)$, then return 0. If $t=\tau(\ell)$, then compute $\sum_{i=\ell}^{n}\Pr[X_0=i]$, or return $\sum_{i=0}^{\ell-1}\Pr[X_0=i]\vartheta(i,\ell,n,t)$ otherwise.

To conclude, it also must be noted that this estimator is accurate provided that the download time is significantly shorter than both the average on- and offline session durations, which occurs when downloading MP3 songs, photos, etc. Indeed, and according to the authors of [32] who conducted a measurement study of Gnutella, around 90% of all files are typically smaller than a few megabytes in size, while only a very small portion of files (around 0.1%) have a size larger than 1 GB. This suggests that our estimator will be accurate in common cases and it agrees very well with our simulation results in Section 6. (See Table 1.)

6. Evaluation

In order to evaluate our analytical framework, we compare the retrieval times estimated in Eq. (6) with the retrieval times obtained by simulating a realistic P2P storage system using availability traces from two real P2P applications. The first traces were obtained by Guha et al. [15] and describe the behavior of 4000 Skype super-nodes during a period of one month. The second traces, obtained by Steiner et al. [29], characterize the behavior of 400,000 KAD peers that were monitored during 6 months. A downside of these traces is the high presence of peers with short lifetime, i.e., the time between the first and last appearances in the trace. Since real storage systems provide incentives to nodes to prolong their membership [14,24], we filtered these traces to use only the nodes with longer lifetimes. We kept the top 1000 nodes with longer membership from Skype traces and the top 10,000 nodes from KAD.

To input our analytical model, we obtained the distribution of online/offline session durations by fitting the Skype and KAD traces to a Weibull variate. Although Steiner et al.

Table 1
Weibull parameters used to fit the session durations. The μ and λ parameters correspond to the scale and shape parameters of the Weibull distribution.

	Online sessions	Offline sessions
KAD	$\mu=0.38,\ \lambda=6300$ $\mathrm{E}[S_1]=6.7$ h	$\mu=0.39,\ \lambda=28,000$ $\mathrm{E}[S_0]=27.7$ h
Skype	$\mu=0.42,\ \lambda=19,000$ $\mathrm{E}[S_1]=15.4$ h	$\mu=0.42,\ \lambda=13,000$ $\mathrm{E}[S_0]=10.5$ h

provided the parameters for a Weibull fitting, we needed to refit after filtering the top 10,000 nodes. Using the result from Yao et al. [40] and as we explained in Section 3, we used the offline session durations of all nodes to obtain S_0, and all the online session durations to fit S_1. In Fig. 7, we can see how the fitted distributions are close to the real distributions of both the online and offline sessions observed in Skype and KAD.

In order to use the Weibull distribution in our analytical framework, we needed to measure the residual lifetimes of the online/offline session durations. We represent the online/offline session duration distributions as S_*. Then, from the Weibull distribution we know that,

$$\Pr[S_* < t] = 1 - e^{-\left(\frac{t}{\lambda}\right)^{\mu}}, \mathrm{E}[S_*] = \lambda\Gamma\left(1+\frac{1}{\mu}\right).$$

And then, using Eqs. (2) and (3), we obtained the residual online/offline distributions, J_*, as:

$$\Pr[J_* < t] = \frac{1}{E[S_*]}\int_0^t (1-\Pr[S_* < u])du = \frac{1}{\lambda\Gamma\left(1+\frac{1}{\mu}\right)}\int_0^t e^{-\left(\frac{u}{\lambda}\right)^{\mu}}du = \frac{\Gamma\left(\frac{1}{\mu}\right) - \Gamma\left(\frac{1}{\mu}, \left(\frac{t}{\lambda}\right)^{\mu}\right)}{\mu\Gamma\left(1+\frac{1}{\mu}\right)}, \quad (7)$$

where Γ is the gamma function, and μ and λ denote the shape and scale parameters of the Weibull distribution,

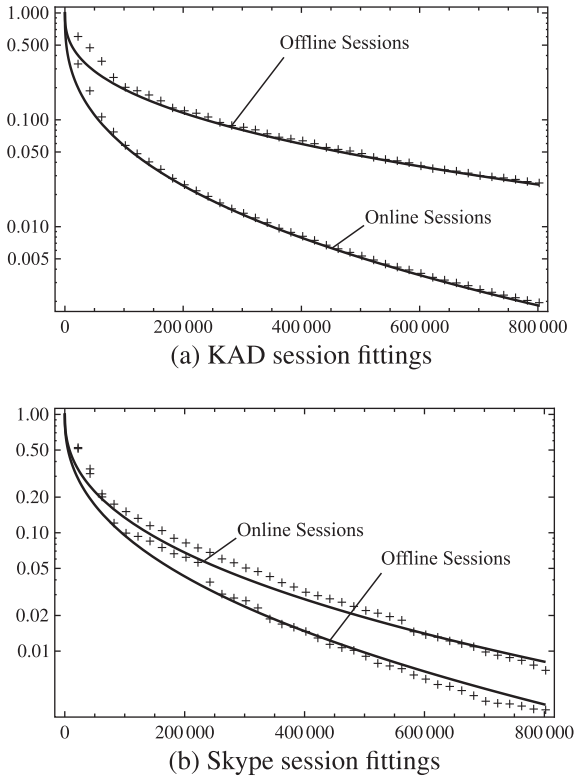


Fig. 7. Log–log plot of the CCDF for the online/offline session durations, S_0 and S_1 respectively, (in seconds) of each trace, and its Weibull fitting. The crosses represent the values obtained from the traces and the continuous line the fitted distribution.

respectively. Due to the mathematical complexity of working with beta and gamma functions, it is practically unfeasible to obtain a closed-form expression of the retrieval time as defined in (6) for Weibull variates. The results presented in this section are then numerical evaluations of all the parts involved in (6).

Although in Section 4 we defined the retrieval time as a generic concept that can be indistinguishably used to represent both reconstruction and repair times, in this paper, we will only consider reconstruction times to evaluate our estimator. For that purpose, we assume a P2P storage system where data is stored using a storage code where the reconstruction degree k is fixed at 30 but for a variable number of storage blocks n, $n > k$, as reported in Table 2. Note that, to achieve a similar level of data availability so that both datasets can be compared with one another, the number of storage blocks n differs significantly, and so does the data redundancy ratio. The reason is that Skype nodes present a higher availability than KAD nodes, for which n can be made clearly smaller.

For simplicity, we assume that the optimal time required to download each individual block, $\tau(1)$, is a system constant determined by the block size α and the upload bandwidth $\omega\uparrow$: $\tau(1) = \alpha/\omega\uparrow$. However, the value of $\tau(1)$ should be short enough to satisfy Assumption 2 —block downloads are not canceled because of node disconnections. This assumption is satisfied when the residual online sessions are much larger than $\tau(1)$, that is, when $J_1 \gg \tau(1)$. To upper bound the value of $\tau(1)$ so that Assumption 2 is shown to hold, one simple way is to define a small positive constant $\epsilon = 0.001$ such that $\Pr[J_1 \leqslant \tau(1)] \leqslant \epsilon$, which loosely speaking means that Assumption 2 is fulfilled with high probability. Given the distribution of J_1 for the KAD and Sype traces, we obtain that the block transmission time $\tau(1)$ must be set to $\tau_{\mathrm{KAD}}(1) = 26$ seconds for KAD and to $\tau_{\mathrm{Skype}}(1) = 59$ seconds for Skype, respectively. Therefore, assuming the number of parallel retrieval processes is $p = 4$, this yields an optimal retrieval time of $\tau_{\mathrm{KAD}}(k) = 208$ seconds for KAD and of $\tau_{\mathrm{Skype}}(k) = 472$ seconds, respectively.

Under this scenario, what will be measured is thus the accuracy of $\widehat{T}(\ell = k = 30, \varphi = \alpha, n)$. We want to note that, although our analytical framework is constrained to Assumptions 1, simulations were not constrained to these assumptions and were conducted by playing back each of the traces for n non-correlated nodes. Concretely, this implied that: (1) there was no restriction in the number of online/offline sessions of each node; (2) block downloads could be canceled because of node departures; and (3) nodes could exhibit daily and weekly patterns. This scenario puts the most pressure on the precision of our estimator when anticipating retrieval times in a real scenario.

In Fig. 8, we show the log–log plots of the complementary cumulative distribution function (CCDF) of 5000 retrieval times obtained by simulation compared with the CCDF derived from our closed-form estimator defined in (6), i.e., $1 - \Pr\left[\widehat{T}(\ell, \varphi, n) \leqslant t\right]$. In the case of the Skype traces where nodes have high online availability (around 60%) the experimental CCDF is closer to the estimated CCDF than in the case of KAD traces (with 20% online availability). One reason for this difference is directly related to

Table 2
Redundancies and availabilities studied.

KAD traces ($a \simeq 0.2$)								
n	110	130	150	170	190	210	230	
Redundancy ratio	$3.\dot{6}$	$4.\dot{3}$	5	$5.\dot{6}$	$6.\dot{3}$	7	$7.\dot{6}$	
Data availability	0.030	0.181	0.476	0.761	0.921	0.981	0.996	
Skype traces ($a \simeq 0.6$)								
n	40	45	50	55	60	65	70	75
Redundancy ratio	$1.\dot{3}$	1.5	$1.\dot{6}$	$1.8\dot{3}$	2	$2.1\dot{6}$	$2.\dot{3}$	2.5
Data availability	0.030	0.200	0.525	0.807	0.945	0.989	0.998	0.999

online availability. Nodes with high online availability present longer online sessions and hence, the cases where Assumption 1 does not hold are significantly reduced. In addition, we can also see how for both traces the difference between the estimated values and the experimental values increase for large values of n, i.e., when the amount of data redundancy increases (recall that k is fixed at 30). The reason is that for large values of n, more retrieval operations finish in optimal time and hence, the absolute error measured using a Kolmogorov–Smirnov test decreases as the value of n grows: *the larger the n value is, the more accurate our estimator becomes*. In general, for a significance level of $\alpha = 0.1$, the Kolmogorov–Smirnov test could not reject the hypothesis that our estimator provides a good fit of the retrieval time distribution in any of the evaluated cases.

To conclude the evaluation we measure how heterogenous bandwidth values affect our estimator. For that purpose, in Fig. 9 we simulate the same Kad and Skype traces for only two values of n. For each node, the simulator picks the time required to download a block from it, $\tau(1)$, uniformly at random from $\{\tau'/2, 3\tau'/4, \tau', 3\tau/2, 2\tau'\}$, where again $\tau' = 26$ for KAD traces, and $\tau' = 59$ for Skype traces. This emulates bandwidth heterogeneity between pairs of nodes. We then compare the retrieval times obtained by simulation (depicted by a line) with the ones obtained by our estimator (depicted by points), evaluated at three possible values of $\tau(1)$. Although this is a simplistic model, it allows us to measure how heterogeneous bandwidths affect retrieval times. We can see from these figures that when $\tau(1) = \tau'$ (which is close to the average download time $\simeq 1.15\tau'$) the estimated CCDF is close to the simulated CCDF. However, the results differ significantly when the value of $\tau(1)$ in our estimator is different than the average value. Although our estimator is designed for systems with bandwidth homogeneity, we can see that when some bandwidth heterogeneity is present, taking the average bandwidth allows to approximate retrieval times with small error.

7. A framework usage guide

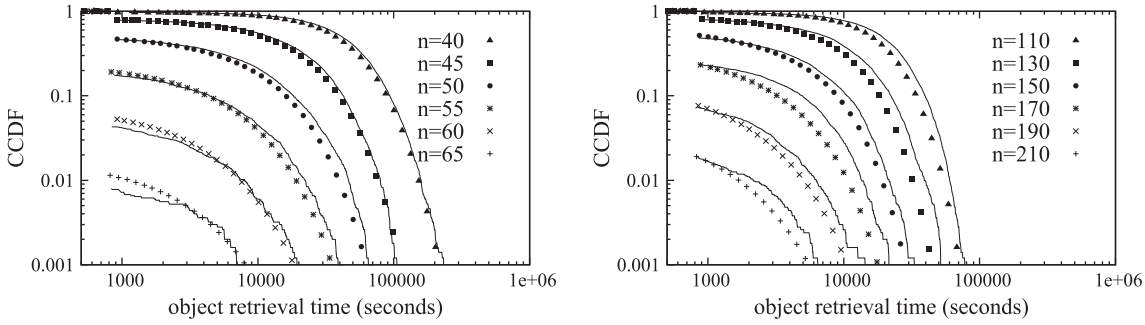
We have evaluated our analytical framework in terms of its accuracy on predicting object retrieval times. Now that we have an accurate estimator of object retrieval times, it is natural to ask how P2P storage systems can take advantage of our framework to allow nodes to trade-off storage costs for retrieval time and optimize their storage systems. For instance, system architects may need to determine to which extent object retrieval times can be lengthened in order to reduce the storage and communication costs. Let us assume a system using a MDS code (MSR Regenerating Code where $k = d$), and that $x \cdot \tau(k)$ is the mean reconstruction time expected by users, being $\tau(k)$ the optimal retrieval time. For a specific x, the system can determine the value of n that achieves a mean retrieval time of $x \cdot \tau(k)$ as:

$$n = min\left\{n' : n' \geqslant k, \mathrm{E}\left[\widehat{T}(k, \alpha, n')\right] \leqslant x \cdot \tau(k)\right\},$$

where the expected retrieval times can be obtained numerically from (6).

Armed with the exact value of n, one can also compute the expected data availability D (probability to detect k out of n blocks online) using the complementary cumulative distribution function of the Binomial variate as:

$$D = \sum_{i=k}^{n} \binom{n}{i} a^i (1-a)^{n-i},$$



(a) Results using Skype fitted parameters (b) Results using KAD fitted parameters

Fig. 8. Log–log plots of retrieval times for KAD and Skype scenarios. The lines depict object retrieval times obtained by simulation when $\ell = 30$ and for different n values. Dots represent the results approximated by $\widehat{T}(\ell, \varphi, n)$ for the same ℓ and n values.

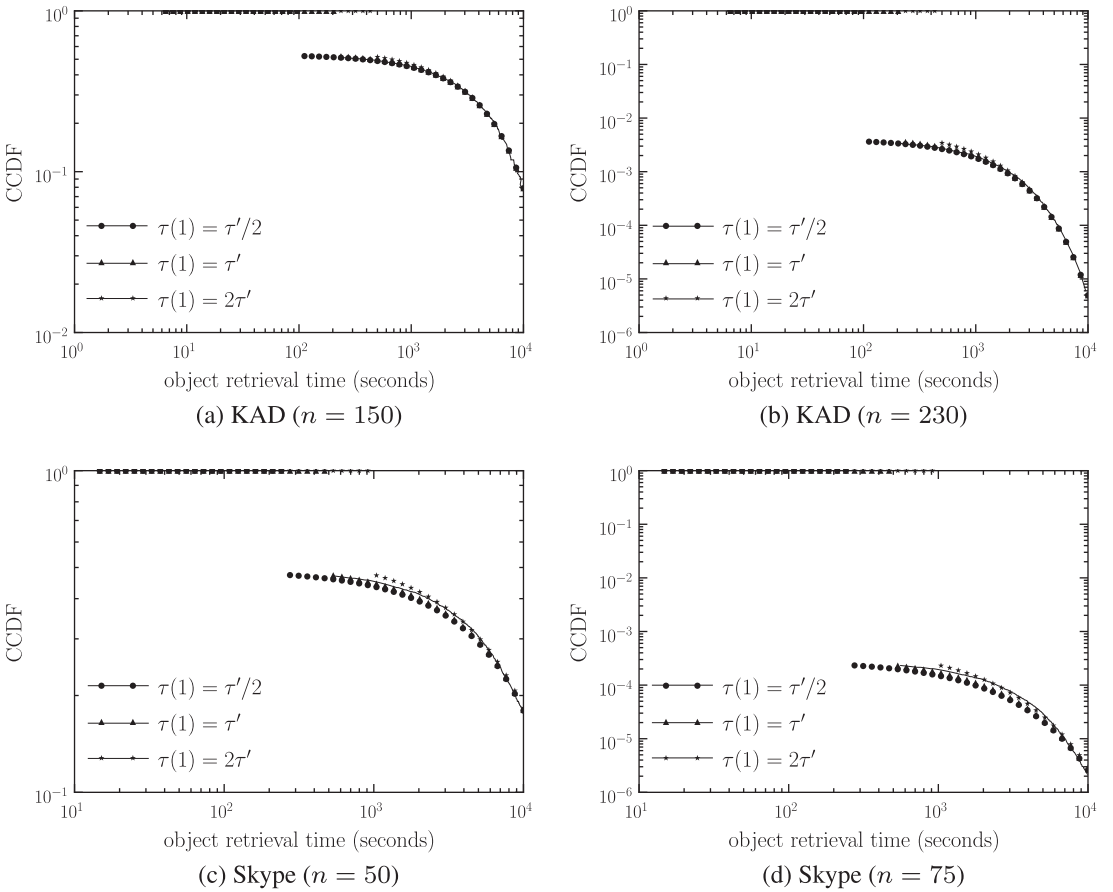


Fig. 9. Log–log plots of retrieval times for KAD and Skype scenarios. The lines depict object retrieval times obtained by simulation when $\ell = 30$ and $\tau(1)$ picked at random from $\{\tau'/2, \tau', 2\tau'\}$. Dots represent the results approximated by $\widehat{T}(\ell, \varphi, n)$ considering different $\tau(1)$ values.

where a is the expected online availability of the storage nodes.

In Fig. 10 we plot for both traces data availability D and the redundancy overhead (i.e., $n\alpha/m$) for nine different expected retrieval times $x \cdot \tau(k)$, where $x \in \{1.01, 1.1, 1.3, 1.5, 1.75, 2, 3, 4, 5\}$. We depict redundancy using squared points and data availability using triangle points. We can see how in both traces, the amount of data redundancy and data availability reduce as the mean retrieval time increases, and such a reduction tends to decrease sharply at the beginning, then steadily beyond certain point. In both datasets, this point equals to 3 times the optimal retrieval time. This suggests that by lengthening *slightly* the object retrieval time, it is possible to reduce data redundancy *significantly* without *excessively* compromising data availability. For instance, in KAD, by setting the mean retrieval time equal to twice the optimal time, data redundancy can be reduced from 8 to 5.7 (a reduction of 28%) while maintaining data availability above 80%. Beyond a certain point, however, further reduction in data redundancy does not compensate the almost linear drop in data availability. This occurs because there exists a threshold in the object retrieval time beyond which data redundancy flattens out, meaning that further reduction in data redundancy will be marginal beyond the threshold but not the drop in data availability.

Using the results from Fig. 10, we can state how storage systems can benefit from our analytical framework and reduce their storage costs:

- **P2P storage systems**: In storage systems like Wuala [36] or OceanStore [18], users trade their local storage resources to obtain an ubiquitous and reliable storage service. The more redundancy required to store each object, the more local resources the user needs to trade. Using our framework, users can individually reduce the amount of local resources they trade by allowing longer object retrieval times: *each user can choose its own trade-off between retrieval times and resources used*. For example, a user that can afford retrieval times 50% longer than optimal retrieval times ($x = 1.5$) can reduce their redundancy requirements by a 30% in KAD traces ($a = 0.2$) and by a 18% in Skype traces ($a = 0.6$). This entails a significant reduction of the amount of traded resources.
- **Backup systems**: In backup systems stored objects are occasionally read. In these environments, repair times should be shorter than the mean time between block failures, i.e. redundant data needs to be repaired faster than it is lost. For example, let us assume that the average lifetime of a node until it permanently disconnects is L. Then, the expected number of blocks that fail during a

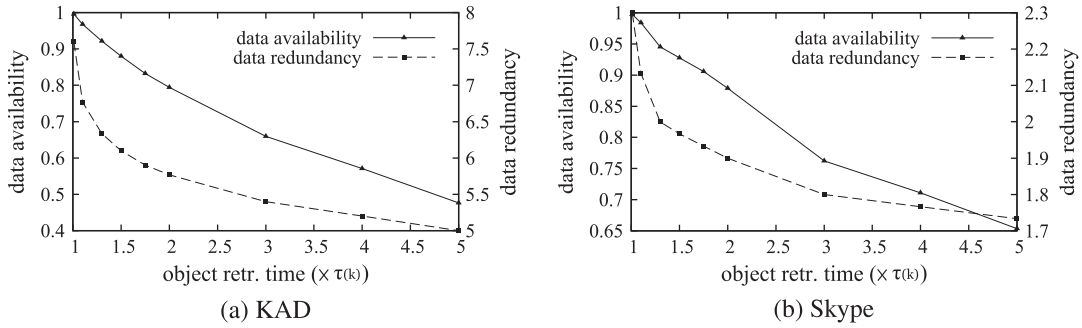


Fig. 10. Redundancy required and data availability obtained to achieve different mean object retrieval times.

repair process is determined by $\mathrm{E}[T(d,\beta,n)]\cdot n/L$ [10], where $T(d,\beta,n)$ is the reconstruction time distribution. Assuming that $\mathrm{E}[T(d,\beta,n)] = x\cdot\tau(d)$, we can determine the number of nodes that fail during a repair as $x\cdot\tau(d)\cdot n/L$. A backup system must configure n to guarantee that $x\cdot\tau(d)\cdot n/L < 1$, i.e. on average, less than one node disconnects during a repair process. For KAD traces, this means that when $x = 5$ and $n = 154$, the storage system is able to repair the lost redundancy provided that mean node lifetime L is at least 44 h. For Skype traces, with $x = 5$ and $n = 56$, the storage system replenishes the lost redundancy if mean node lifetime is $L > 36$ h. Since in storage systems nodes are expected to remain in the system for several months instead of only several hours, lengthening the object retrieval time to 5 times the optimal one can translate into reducing the required redundancy up to 60% without compromising data reliability.

These are some potential applications but we believe that future research will benefit from our estimator to devise, for instance, novel adaptive replication algorithms that are able to reduce redundancy for those objects that are not "hot", i.e., not accessed too much frequently, at the expense of a small increase in the time taken to read them.

8. Conclusions

In this paper, we have introduced an analytical framework to describe the retrieval process in P2P storage systems and proposed one estimate to approximate the retrieval time probability distribution for different data redundancy ratios. The main feature of our framework is that allows the study of the implications that data redundancy and data availability have on object retrieval times. In fact, we have showed that data availability, retrieval times, and data redundancy, are in fact, three faces of a unique storage quality metric. Increasing redundancy always shortens retrieval times and increases availability. On the contrary, reducing redundancy always lengthens retrieval times and reduces availability. Thanks to our framework, we have demonstrated that under real P2P churn scenarios, P2P storage systems can reduce their redundancy up to 60% without affecting more than half of the object retrieval times. By using our framework, P2P storage applications will be able to reduce the storage costs while maintaining optimal service. For P2P backup applications, this means performing maintenance tasks with minimal communication. For other storage systems, it may represent the opportunity to reduce storage and communication costs with an acceptable loss in the retrieval performance.

Acknowledgments

We would like to express our gratitude to the anonymous reviewers for the insights and comments provided during the review process, which have greatly contributed to improve the quality of the original manuscript. This work has been partly funded by the EC FP7 through project CloudSpaces (FP7-317555).

References

[1] D.P. Anderson, Boinc: a system for public-resource computing and storage, in: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (GRID), 2004, pp. 4–10.
[2] C. Blake, R. Rodrigues, High availability, scalable storage, dynamic peer networks: pick two, in: Proceedings the 9th Workshop on Hot Topics in Operating Systems (HOTOS), 2003, pp. 1–1.
[3] S. Buchegger, D. Schiöberg, L.-H. Vu, A. Datta, PeerSoN: P2P social networking: early experiences and insights, in: Proceedings of the Second ACM EuroSys Workshop on Social Network Systems (SNS), 2009, pp. 46–52.
[4] E. Castillo, A.S. Had, N. Balakrishnan, J.M. Sarabia, Extreme Value and Related Models with Applications in Engineering and Science, first ed., Wiley, 2004.
[5] B.G. Chun, F. Dabek, A. Haeberlen, E. Sit, H. Weatherspoon, M.F. Kaashoek, J. Kubiatowicz, Efficient replica maintenance for distributed storage systems, in: Proceedings of the 3rd Conference on Networked Systems Design & Implementation (NSDI), 2006, pp. 4–4.
[6] L. Cox, B. Noble, Samsara: honor among thieves in peer-to-peer storage, in: Proceedings of the ACM Symposium on Operating Systems Principles (SOSP), 2003, pp. 120–132.
[7] L. Cutillo, R. Molva, T. Strufe, Safebook: a privacy-preserving online social network leveraging on real-life trust, Commun. Mag., IEEE 47 (12) (2009) 94–101.
[8] A. Dimakis, P. Godfrey, M. Wainwright, K. Ramchandran, Network coding for distributed storage systems, in: Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM), 2007, pp. 2000–2008.
[9] P. Druschela, A. Rowstron, Past: a large-scale, persistent peer-to-peer storage utility, in: Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HOTOS), 2001, pp. 75 –80.
[10] A. Duminuco, E.W. Biersack, T. En-Najjary, Proactive replication in distributed storage systems using machine availability estimation, in: Proceedings of the 3rd CoNEXT Conference (CONEXT), 2007, pp. 27:1–27:12.
[11] R. Gaeta, M. Gribaudo, D. Manini, M. Sereno, Analysis of resource transfers in peer-to-peer file sharing applications using fluid models, Perform. Eval. 63 (3) (2006) 149–174.

[12] R. Gracia-Tinedo, M. Sanchez-Artigas, A. Moreno-Martinez, P. Garcia-Lopez, F2box: cloudifying f2f storage systems with high availability correlation, in: Proceedings of IEEE 5th International Conference on Cloud Computing (CLOUD), 2012, pp. 123–130.
[13] R. Gracia-Tinedo, M. Sanchez-Artigas, A. Moreno-Martinez, P. Garcia-Lopez, Friendbox: a hybrid f2f personal storage application, in: Proceedings of IEEE 5th International Conference on Cloud Computing (CLOUD), 2012, pp. 131–138.
[14] M. Gramaglia, M. Uruena, I. Martinez-Yelmo, Off-line incentive mechanism for long-term p2p backup storage, Comput. Commun. 35 (12) (2012) 1516–1526.
[15] S. Guha, N. Daswani, R. Jain, An experimental study of the Skype peer-to-peer VoIP system, in: Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS), 2006.
[16] A. Kermarrec, E. Merrer, G. Straub, A. Van Kempen, Availability-based methods for distributed storage systems, in: Proceedings of the 31st IEEE Symposium on Reliable Distributed Systems (SRDS), 2012, pp. 151–160.
[17] R.B. Kiran, K. Tati, Y.-C. Cheng, S. Savage, G.M. Voelker, Total recall: system support for automated availability management, in: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation (NSDI), 2004, pp. 25–25.
[18] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, B. Zhao, Oceanstore: an architecture for global-scale persistent storage, in: Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2000, pp. 190–201.
[19] W.-C. Liao, F. Papadopoulos, K. Psounis, Performance analysis of BitTorrent-like systems with heterogeneous users, Perform. Eval. 64 (9-12) (2007).
[20] W.K. Lin, D.M. Chiu, Y.B. Lee, Erasure code replication revisited, in: Proceedings of the 4th International Conference on Peer-to-Peer Computing (P2P), 2004, pp. 90–97.
[21] J.W. Mickens, B.D. Noble. Exploiting availability prediction in distributed systems. Proceedings of NSDI'06 symposium on networked systems design & implementation. 2006.
[22] F. Oggier, A. Datta, Self-repairing homomorphic codes for distributed storage systems, in: Proceedings of the 30th IEEE International Conference on Computer Communications (INFOCOM), 2011, pp. 1215–1223.
[23] L. Pamies-Juarez, P. Garcia-Lopez, M. Sanchez-Artigas, Availability and redundancy in harmony: measuring retrieval times in p2p storage systems, in: Proceedings of the 10th IEEE International. Conference on Peer-to-Peer Computing (P2P), 2010, pp. 1–10.
[24] L. Pamies-Juarez, P. Garcia-Lopez, M. Sanchez-Artigas, Enforcing fairness in p2p storage systems using asymmetric reciprocal exchanges, in: Proceedings of the 11th IEEE International. Conference on Peer-to-Peer Computing (P2P), 2011, pp. 122–131.
[25] K.K. Ramachandran, B. Sikdar, A queuing model for evaluating the transfer latency of peer-to-peer systems, IEEE Trans. Parall. Distrib. Syst. 21 (3) (2010) 367–378.
[26] I. Reed, G. Solomon, Polynomial codes over certain finite fields, J. Soc. Ind. Appl. Math. 8 (2) (1960) 300–304.
[27] R. Rodrigues, B. Liskov, High availability in dhts: erasure coding vs. replication, in: Proceedings of the 4th International Workshop on Peer-To-Peer Systems (IPTPS), 2005, pp. 226–239.
[28] A.I.T. Rowstron, P. Druschel, Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems, in: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), 2001, pp. 329–350.
[29] M. Steiner, T. En-Najjary, E. Biersack, A global view of kad, in: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC), 2007, pp. 117–122.
[30] D. Stutzbach, R. Rejaie, Understanding churn in peer-to-peer networks, in: Proceedings of the 6th ACM SIGCOMM conference on Internet Measurement (IMC), 2006, pp. 189–202.
[31] D. Stutzbach, R. Rejaie, S. Sen, Characterizing unstructured overlay topologies in modern p2p file-sharing systems, IEEE/ACM Trans. Netw. 16 (2) (2008) 267–280.
[32] D. Stutzbach, S. Zhao, R. Rejaie, Characterizing files in the modern Gnutella network, Multimedia Syst. 13 (2007) 35–50.
[33] L. Toka, P. Cataldi, M. Dell"Amico, P. Michiardi, Redundancy management for p2p backup, in: Proceedings of the 31st IEEE International Conference on Computer Communications (INFOCOM), 2012, pp. 2986–2990.
[34] L. Toka, M. Dell'Amico, P. Michiardi, Online data backup: a peer-assisted approach, in: Proceedings of the 10th IEEE International. Conference on Peer-to-Peer Computing (P2P), 2010, pp. 1–10.
[35] L. Toka, M. Dell'Amico, P. Michiardi, Data transfer scheduling for p2p storage, in: Proceedings of the 11th IEEE International Conference on Peer-to-Peer Computing (P2P), 2011, pp. 132–141.
[36] Various, Wuala, 2010. <http://www.wuala.com>.
[37] X. Wang, Z. Yao, D. Loguinov, Residual-based measurement of peer and link lifetimes in Gnutella networks, in: Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM), 2007, pp. 391–399.
[38] H. Weatherspoon, Design and Evaluation of Distributed Wide-Area On-line Archival Storage Systems. Ph.D. Thesis, University of California, Berkeley, 2006.
[39] H. Weatherspoon, J.D. Kubiatowicz, Erasure coding vs. replication: a quantitative comparison, in: Proceedings of the 1st International Workshop on Peer-To-Peer Systems (IPTPS), 2002, pp. 328–338.
[40] Z. Yao, D. Leonard, X. Derek, X. Wang, D. Loguinov, Modeling heterogeneous user churn and local resilience of unstructured P2P networks, in: Proceedings of the IEEE International Conference on Network Protocols (ICNP), 2006, pp. 32–41.

Lluis Pamies-Juarez is a research engineer at HGST Research. From 2011 to 2013, he was a post-doctoral research fellow in the School of Physical and Mathematical Sciences at Nanyang Technological University (NTU), Singapore. He got his Ph.D. on Computer Engineering in 2011 at Universitat Rovira i Virgili (URV), Tarragona, Spain. His research interests include computer networks, wide-area distributed systems, network coding and coding for storage systems.

Marc Sanchez-Artigas is a tenure-track lecturer at URV (Tarragona, Spain) in the Department of Computer Engineering and Maths. He obtained his Ph.D. degree in 2009 with European Mention at UPF (Barcelona, Spain). His current research focus is on the area of wide-area distributed object technology. He is interested in how to develop new architectures and protocols for P2P networks and Cloud computing, with particular emphasis on the secure and efficient distribution of data, network modeling and reputation systems.

Pedro García-López is a professor in the Department of Computer Engineering and Mathematics at Universitat Rovira i Virgili in Tarragona, Spain. His scientific activity is primarily devoted to computer-supported collaborative work and distributed systems. García has a Ph.D. and M.Sc. degrees in computer science from the University of Murcia, Spain.

Rubén Mondéjar is a research collaborator and a part-time instructor in the Department of Computer Engineering and Mathematics at Universitat Rovira i Virgili in Tarragona, Spain. His research focuses on distributed systems, peer-to-peer networks, and distributed interception middleware solutions. He has a Ph.D., M.Sc. and B.Sc. degrees in computer engineering from Universitat Rovira i Virgili.

Rahma Chaabouni obtained her B.Sc. on Computer Software Engineering in National School of Engineers of Sfax, Tunisia. She is now a PhD student in the Department of Computer Engineering and Mathematics at Universitat Rovira i Virgili in Tarragona, Spain. Under the supervision of Dr. Marc Sanchez-Artigas she is researching on distributed systems and cloud computing. She was before with the ASCOLA team in INRIA, France.