Int J Digit Libr (2012) 12:13–26
DOI 10.1007/s00799-012-0080-5

Extending OAI-PMH over structured P2P networks for digital preservation

Everton F. R. Seára · Marcos S. Sunye · Luis C. E. Bona · Tiago Vignatti · Andre L. Vignatti · Anne Doucet

Published online: 28 February 2012

Abstract Open archives initiative (OAI) allows both libraries and museums create and share their own low-cost digital libraries (DL). OAI DL are based on OAI-PMH protocol which, although is consolidated as a pattern for disseminating metadata, does not rely on either digital preservation and availability of content, essential requirements in this type of system. Building new mechanisms that guarantee improvements, at no or low cost increases, becomes a great challenge. This article proposes a distributed archiving system based on a P2P network, that allows OAI-based libraries to replicate digital objects to ensure their reliability and availability. The proposed system keeps and extends the current OAI-PMH protocol characteristics and is designed as a set of OAI repositories, where each repository has an independent fail probability assigned to it. Items are inserted with a reliability that is satisfied by replicating them in subsets of repositories. Communication between the nodes (repositories) of the network is organized in a distributed hash table and multiple hash functions are used to select repositories that keep the replicas of each stored item. The OAI characteristics combined with a structured P2P digital preservation system allow the construction of a reliable and totally distributed digital library. The archiving system has been evaluated through experiments in a real environment and the OAI-PMH extension validated by the implementation of a proof-of-principle prototype.

Keywords Digital library · Long-term preservation · Digital archiving · Peer-to-peer

E. F. R. Seára (✉) · M. S. Sunye · L. C. E. Bona · T. Vignatti
Department of Informatics, Federal University of Paraná, Curitiba, Brazil
e-mail: rufino@inf.ufpr.br

M. S. Sunye
e-mail: sunye@inf.ufpr.br

L. C. E. Bona
e-mail: bona@inf.ufpr.br

T. Vignatti
e-mail: vignatti@inf.ufpr.br

A. L. Vignatti
Institute of Computing, University of Campinas, Campinas, Brazil
e-mail: vignatti@ic.unicamp.br

A. Doucet
Département DAPA, Université PARIS VI, LIP6, Paris, France
e-mail: anne.doucet@lip6.fr

1 Introduction

As the amount of content in the *World Wide Web* has been rapidly increased during the last years, users are able to rely on particularized tools to navigate through the vast volumes of data, and several information retrieval systems and tools have been proposed to fulfill such a need. The arrival of XML provided the structured organization of contents and made possible a great growth of digital libraries (DL) in several areas.

To create a standard mechanism for information interoperability among DL repositories, the open archives initiative (OAI) [22] proposed the protocol for Metadata Harvesting — OAI-PMH [14]. OAI DL, also known as data providers (DP), are repositories responsible for both storing digital objects and exposing its structured metadata via OAI-PMH framework. Each metadata follows the Dublin Core schema [34] and describes a digital object. Service providers then make OAI-PMH service requests for harvesting metadata and also offers a global search.

The OAI harvesting protocol is compatible with many of open source softwares, like EPrints and Dspace [21,29].

Nowadays, commercial library administration software like Virtua TLS [33] became OAI-PMH compliant and search engines, like Google, are collecting metadata from OAI repositories. An OAI federation is able to offer a global search over a set of DL that share their metadata.

In OAI-PMH each repository stores its digital objects in an independent way. Although this approach keeps the protocol quite simple, because only metadata is shared, it does not rely on digital data preservation, i.e., digital objects can be easily lost if there is no external backup mechanisms assigned to each repository. Moreover, the large necessity for disk space and the cost involved to use specialized hardware makes this solution economically limited. An alternative to digital preservation of content is to replicate the information in multiple storage repositories, which consists on conventional and low-cost computers [5].

Thus, we considered that OAI DL have the same goal, which could be the dissemination of digital content on the Internet. Therefore, OAI might collaborate for the creation of a long-term data preservation environment that ensures reliability and availability of their objects. To enable that collaboration, we proposed an OAI-PMH extension—explained in details in Sect. 3—that keeps current OAI-PMH protocol characteristics and organize DP over a distributed system, allowing to replicate a digital object between nodes in the system.

It is important to note that in our extension, we are interested in replicating only object's content but not its metadata, as the OAI-PMH *harvesting* mechanism performs this job efficiently. However, considering that objects are replicated among DP, which can store content from several other repositories, we created a new OAI-PMH *verb* in the service provider side, called *GetRecord*. This *verb* enables DP to recover any object's metadata in the entire OAI federation.

Besides the storage of replicated objects, our OAI-PMH extension promotes a user-transparent access to them over a Round Robin DNS, which performs requests only to available nodes in the system. With this solution, objects can be recovered from the system even if the original data provider has failed.

Although the proposed extension can be carried out over any distributed system, in as much as it was generically defined, Peer-to-Peer (P2P) networks appear as a promising approach to organize these kinds of systems, as they are highly scalable for the distribution and retrieval of data. At the same time, the available replication mechanisms in most P2P networks are not sufficient to ensure the preservation of data over a long period of time.

To solve this problem, we created a totally distributed P2P archiving system. In this system, OAI repositories are organized by a *distributed hash table* (DHT) [23,26,28,37] and *multiple hash functions* are used as mechanisms for replication. The choice of structured P2P, instead of non-structured, is motivated by its scalability regarding the number of nodes. Moreover, the search algorithms of non-structured P2P networks cannot locate rare items, which is unacceptable in our context. Multiples hash functions are used to perform a selection on specific set of repositories. This is an absent feature in non-structured P2P model using DHT. The system was evaluated through experiments in a real-world environment.

The P2P digital archiving system motivates the definition of a model for data replication, described in Sect. 4. Thus, we propose a model for replication where a reliability metric is associated with each repository. This metric denotes the probability that the data will not be lost or damaged in a given period of time. Furthermore, each item (digital information) needs to be stored with a *desired reliability* that reflects its importance, allowing high or low durability (preservation time). To ensure the desired reliability of an item, several *replicas* of it are created in different repositories. The number of replicas needed to preserve a specific item is determined by the reliability metric of each repository. This allows an optimization of the network resources usage, compared to other systems where the number of replicas is fixed [17,24], however, more elaborated strategies for replication should be used in this case. Another contribution of this study is to present and compare three different strategies for the reliable replication problem.

For example, in Fig. 1, a network with five repositories, identified by 1, 2, 3, 4, and 5; with reliability of 40, 80, 30, 60, and 25% respectively. Suppose that an user wants to insert an item in the network with a *desired reliability* of 90%. A simple strategy would replicate this item in the order of the repositories identifiers, until it achieves the desired reliability. Thus, with one replica in the repository 1, the item has guaranteed a reliability of 40%. Adding a replica in the repository 2, the reliability guaranteed would be $1-(0.6\times 0.2)=88\%$, but still not reaching the desired reliability of 90%. With an additional replica in the repository 3, the guaranteed reliability would be $1-(0.6\times 0.2\times 0.7)=91.6\%$, therefore reaching the desired reliability of the item.

1.1 Contributions and results obtained

This article presents an OAI-PMH extension that defines some new functionalities for both service and DP, to enable

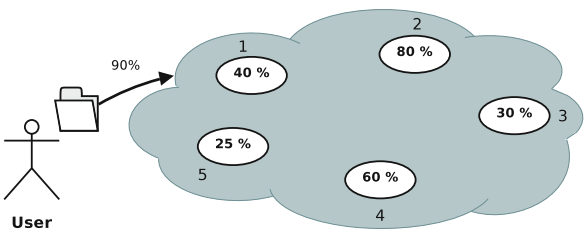

Fig. 1 Repositories labeled with independent reliabilities

digital content preservation in an OAI federation, and uses a Round Robin DNS to promote an user-transparent access to the objects. This extension allows users to see the entire system as a single server, and transparently insert and retrieve objects from the P2P archiving system. Moreover, the proposed OAI extension enables DP to retrieve object's metadata from service providers, making possible to gain access to the object even if its original data provider is crashed. The implementation of a proof-of-principle prototype show a complete integration between OAI extension and P2P archiving system, ensuring reliability and availability in OAI DL.

We also present a model for reliable replication of data in digital archiving systems with immutable data. To cope with the scalability, we create a structured P2P network for the reliable archiving, carried out over the proposed model. We also present three strategies of reliable replication. Experiments in real environments have been made to establish the P2P digital archiving system.

In the proposed model, independent probabilities of failure are associated with each storage repository. These probabilities allow items to be stored based on their preservation time needs. Moreover, the repositories have different storage capacities, i.e., the repositories are *heterogeneous*. Thus, due to limited storage capacity, we aim to insert the maximum number of items in the network; always satisfying the desired reliability of the items.

To maximize the number of items inserted in the network, we designed the strategies with two (heuristic) goals: balance the load among the repositories and, at the same time, minimize the total number of replicas created. This is justified by two facts: (i) balance the load, as well as minimize the replicas created, avoids the overhead in the repositories. Furthermore, (ii) if the load balancing is not performed, a repository can easily becomes completely filled and, consequently, the number of repositories that can be selected decreases, thus decreasing the number of available options that meet the desired reliability of an item. This, in turn, infers the creation of a larger number of replicas to satisfy the desired reliability of an item and hence the total number of items to be inserted in the network decreases due to the limited capacity of the repositories.

In the case of heterogeneous repositories that we consider, balancing the load and minimizing the replicas do not necessarily imply the maximization of the number of items inserted in the network due to different capacities of each repository. Even so, we use these two goals, hoping to approximate the above arguments. Indeed, experimental results of the simulations, presented in [32] confirm that these are good goals for the heterogeneous case.

In Sect. 4.2, we present three strategies for creating replicas. All these strategies aim to optimize the number of replicas created and the load balancing. However, each strategy is based on different arguments to justify its use. The *Randomized* strategy has the load balancing as the main motivation, justified by theoretical results of ball-and-bins [20]. Also, in this strategy, we obtained a theoretical bound in the number of replicas created. The *Greedy over Subset* strategy has the minimization of replicas created as the main motivation, but adjustments are made to perform a good load balancing. The *Ideal Subset* strategy solves the problem without giving priority to minimizing the replicas or balancing the load, acting as a "mean" between the two goals.

Through simulations, presented in [32], we found that among the three strategies presented, the Ideal Subset strategy proved to be the most effective in the creation of replicas and load balancing, and also obtaining the highest number of inserted objects in the network. This strategy has a computational complexity that could not be feasible in practice if not carefully formulated.

Due to high scalability for the distribution and retrieval of data, structured P2P networks appear as natural candidates to implement the model proposed. Thus, in Sect. 5, we present a reliable P2P archiving system, which is one main contribution in this study. The structured P2P networks have a difficulty inherent in the method of routing information which makes difficult the selection of specific nodes (repositories) for the storage. However, this problem was overcome by using *multiple hash functions*, which is the main contribution of this section. We describe in detail the algorithms that use multiple hash functions for the basic operations of the network. The implementation of the P2P digital archiving system was evaluated in a real environment, where digital objects were inserted in a network to stress the preservation time in function of their importance.

1.2 Organization

Section 2 presents some related work and their differences from our study. Section 3 gives a short background about OAI-PMH framework and explain in details the architecture of the OAI extension and how it inserts and recovers object from the system. Section 4 presents the model of replication for archiving systems and the strategies used for creating replicas. In Sect. 5, we present the reliable P2P archiving system and use a real environment for evaluation. The conclusion and future study are presented in Sect. 6.

2 Related work

The usage of P2P networks to create distributed digital libraries has been addressed in several works [2–4,36]. In general, the goal of those approaches is to create mechanisms to search content in distributed repositories.

Freelib [4] is digital library framework based on P2P. It works as a standalone client that once installed on user's

machine, connects itself to a P2P network and automatically create connections among people who share common interests, i.e., have similar searches. Likewise, Flexible CAN [36] is used for information managing and retrieving, and also detecting groups of users with the same interest, creating communities and allowing them to share their resources.

OAI-P2P [3] proposes a P2P network formed by DP to support distributed search over all connected metadata repositories. In this solution, as our study also proposes, DP are organized in a P2P network but, as Freelib and Flexible CAN, no solution to preserve digital content is proposed.

When considering digital preservation and long-term archiving environments, it is very important to emphasize that information is not updated or removed [18], i.e., a replica of an object is *immutable* in the system. Thus, in this study, we are not interested in strategies and mechanisms for replicas update.

Traditional solutions for backup or data storage, such as replicated file systems and RAID disks [15] do not provide a degree of self-management as the P2P networks do. Unlike the proposed P2P archiving system of this study, these systems use a centralized control to manage the content that needs to be replicated. P2P systems for file sharing such as Kazaa, eDonkey, and Gnutella [19] focus on searching resources in dynamic collections, and are not focused on the reliability of the information. In such systems, a file is replicated every time it is copied into a new node.

Other mechanisms, such as CFS [7] and PAST [25] are not able to change the number of replicas dynamically, as they employ replication in the neighborhood. The number of replicas, in fact, cannot exceed the size of the neighbors lists, once the number of neighbors is tied to the DHT protocol. Any digital archiving system built on these systems could not be able to achieve the desired degree of replication required to preserve their information. Other forms of storage that uses DHT as OceanStore [13] and Glacier [11] consider a simpler model where their nodes have equal probability of failure. BRICKS [24] consider availability instead reliability, associating a single fail probability to all nodes in the network.

The lots of copies keep stuff safe (*LOCKSS*) [17] uses a P2P network where nodes are controlled by autonomous organizations to preserve the information for a long period. It uses a complex scheme of audit to detect and repair the damage in the replicas. Unlike our model, the LOCKSS system treats its repositories with a single probability of failure, which does not exactly model real environments. Furthermore, LOCKSS considers a fixed number of replicas for its items and does not integrate with OAI-PMH in a real-time environment.

After studying and evaluating the solutions described above, we believe that our solution has a great difference from the others, which is to join a P2P digital archiving system (with a *desired reliability* for each item inserted) in consonance with OAI-PMH framework, ensuring reliability and availability in OAI DL.

3 Extending OAI-PMH for digital preservation

This section describes an OAI-PMH extension which, together with a distributed archiving system based on P2P networks (Sect. 5), allows OAI-based libraries to replicate digital objects and ensure their reliability and availability. Such extension makes possible communication among DP and allows OAI DL to preserve their digital content with low cost.

To contextualize the reader about the important features in OAI-PMH framework, we briefly present the protocol (Subsection 3.2), and then explain our extension.

3.1 Open archives initiative and protocol

Self-archiving includes storing copies of digital documents on the Internet, to offer open access to it. As this idea has started in 1992, with arXiv System [1], several software have been created to organize the exposure of scientific work on the *World Wide Web* [21,29]. To provide a centralized search among individual archives, the open archives initiative [22] proposed, in 2001, the *Open Archives Initiative Protocol for Metadata Harvesting* (OAI-PMH) [14]. The OAI-PMH framework provides an application-independent interoperability based on *metadata harvesting*, where two classes of participants are found [14]: (i) DP and (ii) service providers.

DP, or OAI repositories, are network accessible servers responsible for storing digital objects and exposing its metadata to harvesters. All DP supply metadata in a common format—the Dublin Core Metadata Element Set [34]—thus can address interoperability and extensibility. Each digital object in a data provider has a digital object identifier (DOI)—in uniform resource identifier (URI) syntax—that is used, in OAI-PMH requests, for identifying the object and extracting its metadata from the repository.

Service providers (SP), or harvesters, are client applications that issue OAI-PMH requests for harvesting either new or out-of-date metadata from a set of DP and store them in a unique database. As shown in Fig. 2, a user access service providers to perform a global search over metadata from an unlimited number of DP. When the user finds the desired metadata in the service provider, a Handle Server maps the data provider where the object is stored from its DOI. Handle servers [30,31] are needed to keep the identifier tolerant to DP changes (server name or even domain name).

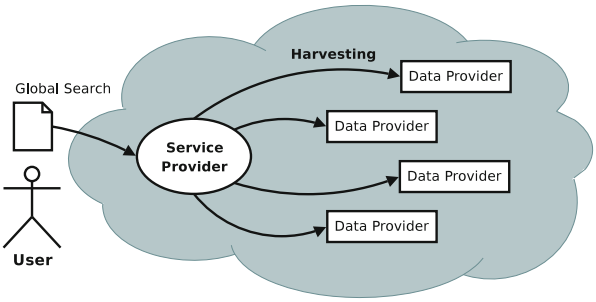

Fig. 2 Global search over a set of OAI repositories

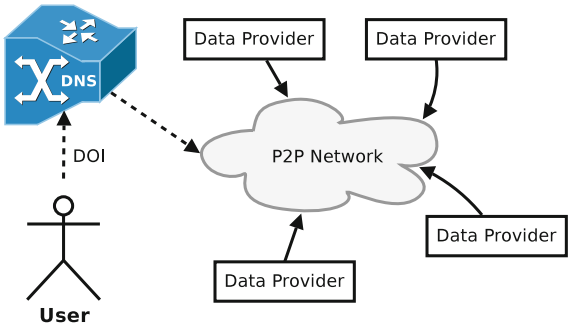

Fig. 3 OAI-PMH extension topology

3.1.1 Problems with this topology

OAI-PMH promotes a very efficient dissemination of metadata on the internet, however, objects are stored in a single data provider, therefore, can become unavailable if this specific computer is *offline*. Moreover, if the computer crashes, content can be permanently lost. To solve these problems, we combine the advantages of OAI-PMH framework and P2P networks to preserve, with low cost, digital content in OAI DL.

3.2 OAI-PMH extension architecture

Many challenges are related to the creation a long-term digital preservation system. In our study, we are mainly interested in keeping digital objects available and reliable for long periods of time, even in the presence of adverse situations.

To achieve this goal, we organize DP as nodes in a P2P digital archiving system, as illustrated in Fig. 3. It means that, despite their original functions (share metadata and store itself content), DP are able to replicate and store replicas of other libraries, reducing the chances of losing the content. Replicas of an object can be stored in several DP, but metadata are maintained and managed only in its original repository, i.e., metadata is not replicated. This decision is justified by the fact that metadata are likely to be modified, and the cost to keep multiple synchronized copies is high.

To ensure that metadata will not be lost and be available whenever we need, our extension requires at least one service provider to regularly *harvest* metadata from all DP in the federation. Thus, a user access the service providers and perform a global search. When a user finds the desired metadata, he gets the DOI and is able to gain access to the object.

It is important to note that in conventional OAI-PMH systems, the DOI normally addresses either a specific data provider or a Handle Server, which redirects to the data provider. If this data provider is *offline*, it is not possible to gain access to the object. To solve this problem, in our extension the DOI does not address to a specific data provider, but to an unique URI that is resolved by a Round Robin DNS (see Fig. 3). This DNS knows which nodes are alive in the network, thus, redirects the user to one of these nodes. When a node receives the request, it shows the object's information and recover the content from the P2P digital archiving system. Requests for any objects can be received from any node.

Next, we show the responsibilities of each component in our extension and how an object is inserted and retrieved in the system.

3.2.1 Data provider

In OAI-PMH framework a data provider is responsible for sharing metadata and providing access to its objects. Our extension defines that a data provider, in addition to its original functions, is part of a P2P digital archiving system, which allows the replication of its objects in other repositories and the collaboration to find objects in any other data provider in the federation.

DP are also responsible for creating both object's DOI and metadata when an object is inserted in the system. The DOI is formed by a unique URI which identifies the federation (stored in a configuration file in the DP), a number that identifies the DP—generated when it is registered in the open archives initiative—and a serial number that locally identifies the object, as presented in Fig. 4. After created, the DOI is inserted in the metadata and is used as *key* to insert and retrieve the object in the archiving system.

In addition, DP are responsible for identifying the *desired reliability* for the object and send this information to the P2P

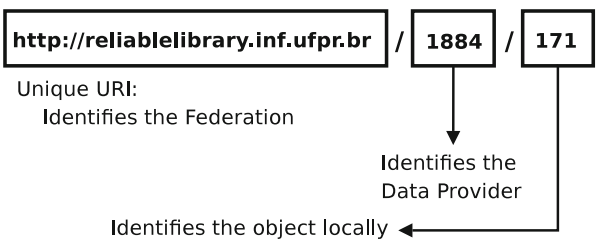

Fig. 4 Digital object identifier structure

archiving system when the object is inserted. This information defines the number of replicas in the system, as presented in Sects. 4 and 5. Finally, if necessary, we extended DP to contact service providers, via the *verb GetRecord*, and retrieve the object's metadata (see Sect. 3.4).

3.2.2 Service provider

In OAI-PMH framework, service providers are responsible for *harvesting* metadata from a set of DP and offer a global search. In our extension, service providers continue performing the same job, however, a group of service providers, called *masters*, should *harvest* all metadata in the Federation. This is needed because in this case the masters are also metadata suppliers to DP. A list identifying the masters is maintained by DP.

To perform this, service providers accept requests via the *verb GetRecord*. This verb follows the same definition of its original, implemented in the data provider side. It receives the DOI as a parameter and returns the corresponding metadata. Requests are made via hypertext transfer protocol (HTTP) and the metadata is delivered following the Dublin Core Metadata Element Set [34].

It is important to note that if either an error or an exception is found, the masters should return a message, in XML format, describing the problem. This message follows the GetRecord specification described in [14].

This solution ensures that metadata is not easily lost, because they are stored in several masters, and allows DP to gain access to metadata when they are not "owners".

3.3 How to insert an object

Inserting a new object in our system is quite simple. First, a data provider reads, from a system administrator, all information that composes the metadata. The content that will be inserted and its desired reliability are also required.

After getting these information, the data provider creates the DOI, following the standard presented in Fig. 4, and inserts it in the object's metadata. Once created, the metadata can be *harvested* by service providers.

Finally, the data provider invokes the function `insert`, implemented in the P2P digital archiving system, whose replicates the content over the network until reaches its *desired reliability*. The Algorithm 1 illustrated those steps.

```
input: metadata_info, content, reliability
begin
    DOI ← URI + DPNumber + sequence  /*Creates the DOI          */
    metadata = metadata_info + DOI
    P2P.insert(DOI, content, reliability)
end
```

Algorithm 1: Insert by Data Provider

3.4 How to retrieve an object

The first step to retrieve an object from the system is to access a service provider and recovers its DOI. As a DOI is a simple URI, users can use a web browser to obtain information about the object and access its content.

To address a specific computer, a URI is normally resolved by a domain name system (DNS). In our extension, a Round Robin DNS is used to perform this. A Round Robin DNS [6] is a technique of load balancing, or fault-tolerance for redundant internet protocol service hosts (e.g., web servers) by managing the DNS responses to address requests from client computers according to an appropriate statistical model.

There are many implementations of Round Robin DNS, in its simplest form, it works by responding to DNS requests not only with a single IP address, but with a list of IP addresses of several servers that host identical services. Thus, when a user access the DOI he is redirected to an available data provider. Available nodes are maintained by a *node checker*, which regularly checks the nodes activity. If a node is *offline*, it is removed from the list until recover its normal operation.

When a data provider receives the request for an object, it verifies if the object's metadata is locally stored. If it is not, the data provider requests the metadata for the service provider, via *GetRecord verb*, using the DOI as input. Once the metadata is recovered, the data provider presents its information to the user and recovers the content from the P2P digital archiving system using the DOI as *key*, as illustrated in Algorithm 2.

```
input: DOI
begin
    metadata = get_metadata_local(DOI)
    if metadata not found then
        metadata = ServiceProvider.getRecord(DOI)
        if metadata not found then
            return "Object does not exist"
    /*shows object's information to the user                    */
    showMetadata(metadata)
    return P2P.retrieve(DOI)
end
```

Algorithm 2: Retrieve by Data Provider

3.5 Integration tests

To evaluate our OAI extension, we implemented a proof-of-principle prototype. This prototype was implemented in Java programming language and XML documents were used to store metadata and information to configure the system. Messages between the components were sent and received by Sockets [35].

The main purpose of these tests is to show that OAI-PMH extension works in together with P2P digital archiving system, explained in Sect. 5. To achieve this goal, we create a set

```
Retrieve process starting.
DOI received: http://archive.local/1884/196
Looking for a local metadata
 - Metadata not found
Asking Service Provider for metadata (GetRecord)
 - Metadata retrieved
Information presented to the user
 - title: Everton's thesis
Asking DHT for digital object
 - Object retrieved
Retrieve process finished.
```

Fig. 5 Data provider log: illustrate an object being recovered

of *logs* during the execution of Algorithms 1 and 2. Figure 5 shows a *log*, from a data provider, which illustrates an object being recovered.

It is important to note that we mapped several different behaviors with logs. In [8], it is possible to find other experiments that helped us to validate our proposed extension. In a real-world environment, it can be implemented over a number of open source softwares, like EPrints and Dspace [21,29].

4 Data replication model and proposed strategies

The model is composed by a set N of $|N| = n$ items (digital objects). All items have identical size, without loss of generality, equal to 1. Each item i has associated a probability $0 < r_i < 1$, called the *desired reliability* of the item. Furthermore, we have a set $M = \{\ell_1, \ldots, \ell_m\}$ of $|M| = m$ repositories, where each repository ℓ_j has associated a *storage capacity* c_j and a probability $0 < p_j < 1$, called the *reliability* of the repository. To determine this reliability, we can consider some parameters, such as the number of bugs, the vulnerabilities of the system, human factors, the frequency at which the machines are repaired, among others. The determination of the desired reliability of an item is not in the scope of this study.

The reliability of the subset $S \subseteq M$ is defined as $1 - \prod_{\ell_j \in S}(1 - p_j)$, which denotes the probability of at least one repository in S does not lose data in a given time interval.

We define the problem as follows. Items arrive one by one, i.e., initially there is no item, and the items arrive as time passes. After an item i arrives, we have to choose a subset S_i of repositories where each repository in S_i receives a copy of the item (*replica*) to satisfy the desired reliability r_i. In other words, we select $S_i \subseteq M$ such that

$$1 - \prod_{\ell_j \in S_i} (1 - p_j) \geq r_i. \tag{1}$$

In addition, each repository in S_i must have enough free space to receive the copy. The *load* of a repository is the number of replicas assigned to it. The replicas are never updated, i.e., they are immutable. The objective of the problem is to maximize the number of items inserted in the network, satisfying the desired reliability of items and the capacity of the repositories.

As argued before, we focus on two goals: (i) on the one hand, we want to minimize the total number of replicas created, i.e., minimize $\sum_{\forall i} |S_i|$. On the other hand, (ii) at the same time, the replicas should be placed so as to balance the load. To measure the load balancing, we evaluate two metrics: the *makespan*, i.e., the load of the most loaded repository, and *standard deviation* of the repositories load. Note that the minimization of the makespan and the standard deviation of the loads are sufficient measures to ensure that all repositories have a balanced load.

In what follows, we make some comments regarding to the feasibility of the problem

Observation 1 *Assigning two or more replicas of the same item to a repository does not bring any benefits.*

By Observation 1, in the worst case we create m replicas for each item.

Observation 2 *There are instances which are not feasible solutions to the problem.*

To illustrate Observation 2, suppose that all repositories have a low reliability, e.g., $1/(m+1)$. By Observation 1, we do not take advantage in creating more than m replicas. Thus, when creating replicas of a given item in all m repositories, we guarantee the following reliability.

$$1 - \left(1 - \tfrac{1}{m+1}\right)^m \leq 1 - \tfrac{1}{e} \leq 2/3.$$

In other words, it is impossible to guarantee a desired reliability of an item that requires a reliability greater than 2/3. In view of Observation 2, we would like to identify conditions that make an instance of the problem feasible/infeasible. A necessary and sufficient condition (i.e., an "if and only if" condition) would be ideal in this case.

Observation 3 *Due to the on-line nature of the input, it is impossible to decide if an instance of the problem is feasible/infeasible.*

By Observation 3, we can only say that an instance is infeasible if the item that came last makes it infeasible, excluding the existence of a necessary and sufficient condition for detecting feasibility. However, without "looking into the future" and look only at the items that already arrived, the problem is feasible if and only if $1 - \prod_{\ell_j \in S_i}(1 - p_j) \geq r_i$, $\forall i \in N$.

A naive approach would treat both goals in an independent way, for example, first solving the minimization of the replicas and then balancing the load. This is not a good approach,

because the number of replicas created depends directly from repositories which had allocated the replicas. As an example, we illustrate a situation where we first solve the problem of replicas and then solve the load balancing. Suppose an instance where the repository ℓ_j has reliability p_j and all items have desired reliability less than p_j. It suffices to create a single replica of each item in ℓ_j. However, ℓ_j will be overloaded, and when we start the phase of balancing the load, we have to remove items from ℓ_j. In this way, we have to select other repositories to accommodate new replicas of these items to meet their desired reliability, lying again on the problem that we thought was solved before the load balancing.

4.1 Equivalent definition

Next, we rewrite the desired reliability constraint to ease the handling of the problem of replicas creation. We present an equivalent definition of the desired reliability constraint, replacing the product by a summation. We rewrite the desired reliability constraint as follows:

$$\begin{aligned} 1 - \prod_{\ell_j \in S_i} (1 - p_j) \geq r_i &\equiv \prod_{\ell_j \in S_i} (1 - p_j) \leq 1 - r_i \\ &\equiv \prod_{\ell_j \in S_i} e^{\ln(1-p_j)} \leq e^{\ln(1-r_i)} \\ &\equiv e^{\sum_{\ell_j \in S_i} \ln(1-p_j)} \leq e^{\ln(1-r_i)}. \end{aligned}$$

But

$$e^{\sum_{\ell_j \in S_i} \ln(1-p_j)} \leq e^{\ln(1-r_i)} \iff \sum_{\ell_j \in S_i} \ln(1 - p_j) \leq \ln(1 - r_i).$$

As $0 < p_j, r_i < 1$, then the value of the logarithm function is negative. Thus, for clarity of notation, we redefine the problem variables. Let $a_j = -\ln(1 - p_j)$ and $b_i = -\ln(1 - r_i)$. Therefore, the desired reliability constraint can be rewritten in the following equivalent way,

$$\sum_{\ell_j \in S_i} a_j \geq b_i. \tag{2}$$

That is, for an item i we have to select $S_i \subseteq M$ such that the sum of a_j exceeds b_i. Equation 2, besides being easier to handle, is equivalent to the Eq. 1.

4.2 Strategies for replicas creation

In what follows, we present three strategies for the creation of replicas. In all strategies, we do a selection over a set of repositories. When doing a selection on M, the complexity of the worst case is linear in m, which is a good theoretical bound. In practice, however, a selection on the set of all machines of the network is infeasible. Thus, in the strategies described below, we denote by $M^{\text{o}} \subseteq M$ the set of machines that are available based on a feasible number of machines that we can select in practical situations. For each item that arrives to be inserted, we consider that M^{o} is selected at random from M; in practice, the way that M^{o} is selected may depends on the system features. Note that the strategies presented are generic, i.e., can be used in various situations of reliable replication systems. Therefore, the size of M^{o} may depends on the features of the real-world situation that we are considering.

It is worth noting that in all strategies, when a replica is assigned to a full repository, we ignore it and randomly select another repository of the considered set.

4.2.1 Randomized strategy

Here, we present the first proposed strategy to solve the problem. Algorithm 3 shows the details.

```
begin
    S = ∅
    while reliability of S is less than r_i do
        choose ℓ_j ∈ M^o uniformly at random
        S = S ∪ {ℓ_j}
    return S
end
```

Algorithm 3: Randomized

In this subsection, we use $m = |M^{\text{o}}|$ and assume that the reliabilities of the repositories are uniformly distributed in an interval. Formally, the values p_j is chosen according to the *continuous uniform distribution* in the interval $[a, b]$, where $a > 0$ and $b < 1$. We assume that the expected number of reliable repositories (in a specified time interval) is at least $8 \ln m$; note that this is not a strong assumption to the problem, as the expected fraction $\frac{8 \ln m}{m}$ of reliable repositories goes to 0 when m goes to infinity. For example, for $m = 1{,}000$, we assume that the expected fraction of reliable repositories is $\frac{8 \ln 1{,}000}{1{,}000} \approx 0.05$, i.e., we assume that, in expectation, 5% of repositories are reliable.

Let X_j be a random variable that is equal to 1 if the repository ℓ_j is reliable in the time interval specified, 0 otherwise. Let $X = \sum_{j \in M^{\text{o}}} X_j$ be the random variable of the total number of reliable repositories in a given time interval. As we assume that the reliabilities of the repositories are distributed according to the continuous uniform distribution, then $E[X] = \frac{m(a+b)}{2}$. In a given time interval, X can be less than $E[X]$, however, with high probability, it cannot be much less than the expected value, as shown in Lemma 1.

Lemma 1 *Let X_j be a random variable that is equal to 1 if the repository ℓ_j is reliable, 0 otherwise. Let $X = \sum_{\ell_j \in M^{\text{o}}} X_j$ be the random variable of the total number of reliable repositories in a given time interval. Then $Pr(X < \frac{1}{4} E[X]) < \frac{1}{m^2}$.*

Proof Note that X is a sum of Poisson random variables. Therefore, we can apply a known Chernoff bound [20], obtaining $\Pr(X < (1-\frac{3}{4})E[X]) \leq e^{-\frac{9E[X]}{32}} \leq m^{-2}$, where the last inequality follows from the fact that $E[X] \geq 8\ln m$.

Lemma 1 tells us that with high probability (i.e., probability greater than $1-\frac{1}{m^2}$), a fraction of $\frac{a+b}{8}$ of the repositories are reliable. That is, with high probability, when selecting a repository uniformly at random, we have probability at least $\frac{a+b}{8}$ that it is a reliable repository. Thus, by choosing k repositories uniformly at random, the probability that one or more repositories are reliable is at least $1-(1-\frac{a+b}{8})^k$ and we want that this probability be greater than the desired reliability r_i. Thus, by choosing $k = \lceil \frac{8}{a+b} \ln(\frac{1}{1-r_i}) \rceil$ repositories uniformly at random and place the replicas on them, the desired reliability of the item i is satisfied with high probability, as the Theorem 1 claims.

Theorem 1 *If item i places $k = \lceil \frac{8}{a+b} \ln(\frac{1}{1-r_i}) \rceil$ replicas in k repositories chosen uniformly at random then, with high probability, the desired reliability of i is satisfied.*

As an example of Theorem 1 application, suppose that the reliability of the repositories are uniformly distributed between 50% and something close to 100%. Thus, an item with desired reliability of 95% needs to choose a minimum of $\lceil \frac{8}{0{,}5+1} \ln(\frac{1}{1-0{,}95}) \rceil = 16$ repositories uniformly at random to place its replicas.

Regarding the load balancing, we note that in Algorithm 3, a replica always chooses a repository uniformly at random. That is, Algorithm 3 simulates the balls-and-bins process, well studied in the area of randomized algorithms and the existing results can be used in our problem [20]. The results of Theorems 2 and 3 can be easily obtained from the results of balls and bins and therefore the demonstrations will be omitted.

Theorem 2 *If n balls are thrown independently and uniformly at random on m bins, then the load of the most loaded bin is bounded by $2e\frac{n}{m} + 2\log m$ with high probability.*

Theorem 3 *Let $n \geq m \log m$. If n balls are thrown independently and uniformly at random on m bins, then the load of the most loaded bin is bounded by $\frac{n}{m} + \sqrt{8\frac{m}{n}\log m}$ with high probability.*

Theorems 2 and 3 are related, respectively, to a small and a large number of balls (in our problem, they are the items). However, in practical situations, it is likely that $n \geq m \log m$ and, in this case, Theorem 3 tells us that with high probability the most loaded repository have a constant number of items in addition to the optimal balance.

4.2.2 Greedy over subset strategy

Suppose we want to minimize the total number of replicas without worrying about the load balancing. Thus, using the equivalent definition of Sect. 4.1, it suffices to solve the integer linear program (ILP) below

$$\min \sum_{\ell_j \in M^o} x_j$$

$$\text{s.t.} \sum_{\ell_j \in M^o} a_j x_j \geq b_i$$

$$x_j \in \{0,1\} \qquad \forall \ell_j \in M^o.$$

The ILP described can be optimally solved by sorting the values a_j in non-increasing order and taking the values in such order until the sum of the selected values reaches b_i. This greedy strategy, besides being very simple and efficient, optimally solves the ILP as, in a given step, we have no advantage in selecting a value less than the value in the sorted order. Algorithm 4 shows the details of this strategy.

```
begin
    sort M^o in non-increasing order according to the values a_j
    S = ∅
    t = 1
    while Σ_{ℓ_j ∈ S} a_j < b_i do
        S = S ∪ {ℓ_t ∈ M^o}
        t = t + 1
    return S
end
```

Algorithm 4: Greedy over Subset

Note that this strategy accumulates the replicas on the repositories in M^o with higher reliability, which is not good for the load balancing. We know that M^o is randomly chosen in each insertion of an item, but this do not suffices to improve the load balancing because if M^o is large, we lie again in the problem of accumulating the replicas in repositories with higher reliability. Moreover, if M^o is small, we do not have many options to choose or there are not enough repositories to satisfy the desired reliability of an item. Several sizes of M^o are evaluated in [32].

4.2.3 Ideal subset strategy

To create the replicas, we select the subset $S \subseteq M^o$ that provides the reliability that is closest to the desired reliability of the item. That is, we choose $S \subseteq M^o$ that minimizes $1 - \prod_{\ell_j \in S}(1-p_j) - r_i$. Using the equivalent definition of the problem, we need to solve the following ILP

$$\min \sum_{\ell_j \in M^o} a_j x_j - b_i$$

$$\text{s.t.} \sum_{\ell_j \in M^o} a_j x_j \geq b_i$$
$$x_j \in \{0,1\} \qquad \forall \ell_j \in M^o$$

Note that if the solution value is equal to 0 then there is a subset of values that together sums exactly b_i; if the solution value is greater than 0, then there is no such subset. Thus, if we solve this ILP, then we could also solve the SUBSET-SUM decision problem, which is NP-complete [9]. Therefore, as the ILP is an optimization problem, then it is NP-hard; assuming P$\neq$NP, no polynomial algorithm can solve such problem. However, there is a *dynamic programming* algorithm which solves this problem in pseudo-polynomial time, but in practice is satisfactory for the vast majority of instances [9]. The details of the algorithm are omitted.

The solution of the above ILP do not necessarily provides a good solution to minimize the number of replicas created. Nevertheless, the ideal subset strategy is motivated by the fact that in practical situations it is expected to not create too many replicas and to select different subsets for each item, so that the distribution of replicas in the repositories balance their loads. Note that, if we had not used the equivalent definition, we are not able to model this case as a subset sum problem, and then we need a real exponential algorithm to solve this, which turns out to be an unfeasible alternative to this case.

5 A peer-to-peer digital archiving system

The model proposed in Sect. 4 is designed in a generic way and can be implemented on any distributed mechanism for organizing the storage repositories. In particular, structured P2P using DHT appears as natural candidate as it is highly scalable for data distribution and retrieval. However, a difficulty inherent in structured P2P networks is the accurate selection of nodes (repositories) to store the replicas. It is not trivial to select a specific subset of nodes using the routing method from DHTs. Therefore, the implementation of the digital archiving system needs to define an architecture that accommodates all the features that the model of Sect. 4 requires. Thus, we present a scheme of selection of nodes using *multiple hash functions*, which allows the selection of a particular set of nodes.

5.1 Architecture

5.1.1 Structured P2P networks and DHT

The system routes the messages of the network through structured P2P networks, using DHTs. The choice of structured P2P, instead of non-structured, is motivated by its scalability regarding the number of nodes. A problem of non-structured P2P networks is that they often use brute force algorithms to perform the search ("flooding") and are more suitable for popular content. Moreover, in many cases, the search algorithms of non-structured P2P networks cannot locate rare items, which is unacceptable in the context of digital archiving where the objects are equally popular [16].

DHTs have a problem with the transient population, i.e., maintaining the structure of the routing tables is relatively expensive in *churn* situations. However, the machines transiency in organizations that intend to preserve digital documents is not as frequent when compared with machines used in traditional applications in the non-structured architecture [17]. Therefore, the necessary adjustments in the topology of the structured networks does not overload the archiving system in case of churn.

5.1.2 Specific selection of repositories

The strategies proposed in Sect. 4.2 assume that is possible to do a selection on specific repositories. However, the DHT by itself does not provide the mechanisms of selection of a specific node, due to its method of index keys. So if we are interested to store the content in a given set of repositories, we must provide a mechanism that simulates the process of specific selection. To perform the selection of specific nodes, we propose the use of *multiple hash functions*, as explained below.

A digital object consists of a *key*, which is the identifier of the object; a *value*, which is the content of the object; and a parameter of *desired reliability*, which is the reliability that should be achieved when inserting the object in the network. Let $h_1, h_2, \ldots, h_r$ be the r *hash functions*. The hash functions have global visibility, i.e., they are the same for all nodes. Given the key k of a digital object, we apply k to the hash functions, i.e., $h_1(k), h_2(k), \ldots, h_r(k)$. Each of the r generated hash maps to a node in the network. Thus, for each object to be inserted, we get r nodes where we can place replicas (i.e., the set M^o). ¿From this set of r nodes, we use a strategy of replica creation (e.g., the strategies of Sect. 4.2) to define the subset of these r nodes that receive the replicas. It is not difficult to obtain a family of such hash functions. One way is to use a single hash function h and append a number $i = 1, \ldots, r$ to the key of the object, which is used as argument to the function h. For instance, if the object key is the string *foo*, then $h(foo1), h(foo2), \ldots, h(foor)$ would give us r hashes of this object.

Figure 6 illustrates the selection of repositories which are performed by three different objects. In the figure, keys are represented by `object_a`, `object_b`, and `object_c`, and the dotted lines denotes the set of nodes associated with each key after applying $r = 6$ hash functions. As we have 6

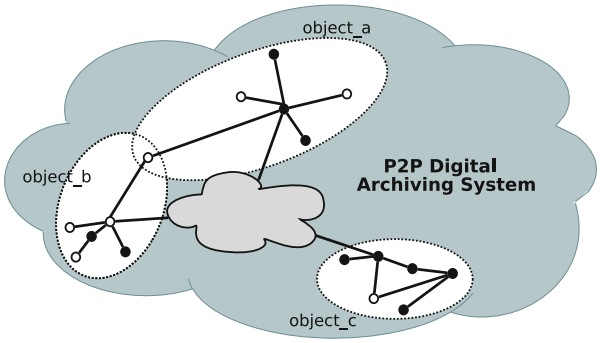

Fig. 6 Subsets of repositories associated with their respective digital objects

hash functions, the resulted hashes maps to 6 nodes of the network. From each subset of nodes associated with an object, the strategy of creating replicas is applied to determine the nodes that receive the replicas. The black circles represent the repositories that have been chosen to put the replicas.

It is worth noting that performing a selection among *all* nodes of the network (or equivalently, using m hash functions) is not a good approach because an instance would have the size of the network, which is unfeasible in real-world situations. On the other hand, considering a subset of small size is a feasible option of implementation even in face of the scalability of the network. Thus, based on the results showed in [32], we can choose the optimal size of the subset and, therefore, decide how many hash functions to use.

The main advantage of using multiple hash functions is the ability to select specific nodes. Without this, the strategies proposed in Sect. 4.2 could not be implemented in structured P2P networks using DHT. Moreover, an advantage of using the strategy of multiple hash functions is the ability to use any DHT protocol to route messages in the network, unlike the strategies for replication in P2P using the *neighborhood* or the *path* [12], which are tied to the protocol. Multiple hash functions also allow flexibility for digital objects to have different numbers of replicas, which is not possible in the *symmetric* strategy of replication [10]. Another feature is the easy retrieving of a given replica without necessarily retrieve another one previously, making it easy to develop a retrieving algorithm. In the *correlated hash* strategy [12], all the keys of a given object are correlated with the first key, thus not allowing this feature.

The selection of repositories proposed above involves no centralized information about the location of the stored replicas. An alternative would be the use of a super-node that had a "directory" which could be consulted about the information of the exact location of replicas of a given object. However, this super-node would be a contention point. Our system avoids super-nodes, adopting a completely distributed approach where no information is centralized.

5.2 Algorithms for system operation

To operate the P2P digital archiving system, we need to define some basic operations of the system. In this section, we present the algorithms `insert` and `retrieve`, used respectively for the insertion and retrieving of a digital object in the system. These algorithms implement multiple hash functions discussed earlier. Note that both algorithms are executed locally in each node. For example, a user who wishes to insert or retrieve an object contacts any node of the network, which in turn initiates the process of routing the message of the DHT.

```
input: key, value, reliability r_i
begin
    M^o = ∅
    for i = 1 to r do
        M^o = M^o ∪ {ℓ_{h_i(key)}}
    S = insertion_strategy(M^o, r_i)
    foreach s ∈ S do
        j ← hash function number of s
        put(h_j(key), value)
end
```

Algorithm 5: insert (key, value, reliability)

To insert an object, the routine `insert(key, value, reliability)` is used, as shown in Algorithm 5. When inserting an object in the network, the desired reliability of the object is previously chosen by the user. Initially, M^o starts empty. The first loop selects the subset M^o of size r associated to the key of the object; $\ell_{h_i(key)}$ is an abuse of notation which denotes the node pointed by the ith hash of the key. In this loop, we implicitly save the values i used for each node; this will be used later. After that the function `insertion_strategy` (M^o, r_i) is executed, which returns the subset $S \subseteq M^o$ of nodes that will receive the replicas. The function `insertion_strategy` can be replaced by any strategy of replication, for example, those presented in Sect. 4.2. In our implementation, the reliability of the nodes are stored in the DHT. The last loop is where the insertion occurs. The value j denotes the hash function number of the object s considered; as stated previously, these values were saved in the first loop. The routine `put[`h_j`(key), value]` puts a replica of the object in the chosen location.

The implementation of the algorithm takes care of not assigning more than two replicas of the same object in one node.

To retrieve the object, the user performs the function `retrieve(key)`, where `key` is the key of the object to be retrieved, as shown in Algorithm 6. The idea of the algorithm is to search in all r nodes for a replica of the item, using the hash functions for that. There are two cases where the DHT does not return the object: when the node is not present or the node does not contain a replica of the object.

```
input: key
begin
    for i = 1 to r do
        value ← get(h_i(key))
        if value is not null then
            return value
    return −1  /*not found                    */
end
```

Algorithm 6: retrieve (key)

5.3 Experimental results

The P2P archiving system was implemented and evaluated through experiments carried out in a real-world environment. The implementation uses the *Overlay Weaver* environment to build networks [27]. Overlay Weaver provides great flexibility in the choice of DHT protocols and other high-level services implemented as overlay networks. In particular, this environment supports Chord, Pastry, Tapestry, and Kademlia. In our experiments we use Chord. It is worth noting that Overlay Weaver has itself a mechanism of replication that has been turned off for the evaluation of our experiments. The experiments were conducted in a network with 12 nodes; this quantity is enough to validate the ability of long-term archiving and the mechanism of creation of replicas.

The reliability of each node was considered to be the probability of the node do not lose information during the period of 1 year. Thus, we can simulate the state of the network regarding the preservation of information over the years. In our experiments, we do not evaluate changes in the reliability of the nodes over the years.

The experiment starts with a network where various objects are inserted. Objects are inserted with different reliabilities. For the replication of objects we used the ideal subset strategy, with the size of the subsets equal to 6. Initially, all nodes are reliable (*online*). When a node becomes unreliable, it loses its information and is disconnected from the network (*offline*). We purposely did not implement a strategy to recover *offline* nodes because we want to stress that objects with high desired reliability are preserved longer.

Figure 7 shows the results. Of the total of 12 nodes, 2 of them have 30% of reliability, 2 have 50%, 2 have 70%, 3 have 80%, and the remaining 3 have 90% of reliability. Initially, in the first year, we include a total of 25 objects, divided into 5 groups, each group containing objects with desired reliability respectively, 30, 50, 70, 90, and 99%.

After the first year of existence of the system, a node of reliability equal to 30%, a node with 50% and another with 90% failed. Even with these fails, all the 25 objects were able to be retrieved. At the end of the fifth year of the system, two nodes (30 and 70% of reliability) get disconnected from the network. In the fifth year, it was not possible to retrieve the 3

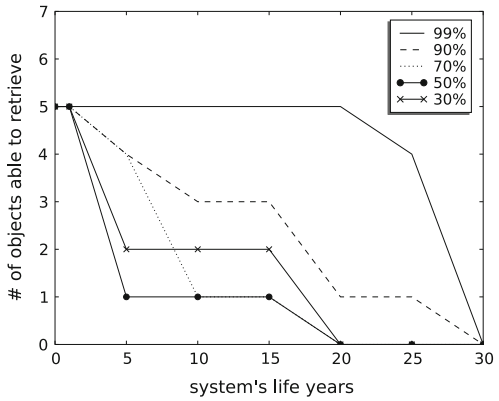

Fig. 7 Number of objects able to retrieving information depending on the age of the system

objects with 30% of desired reliability, 4 objects of 50%, 1 of 70%, and 1 of 90%. In the tenth year, three additional nodes gets *offline*. They are nodes with reliability of 70, 80, and 90%, respectively. It was unable to locate a total of 3 objects of 30% of desired reliability, 4 objects of 50%, 4 of 70%, and 2 of 90%. At the end of the fifteenth year, a node with 80% of reliability was disconnected and the same objects of the tenth year were possible to be retrieved. In the twentieth year, 1 node of 50% was disconnected and only one node of 80% and another of 90% left the system. In this age of the system, it was able to locate all objects of 99% of desired reliability and 1 object with 90%; all other objects were lost. In the twenty-fifth year, the node of 80% failed and it was still possible to retrieve 4 objects of 99% and 1 of 90%. In the thirtieth year, the last node were disconnected from the network and no further replicas existed.

6 Conclusions and future study

The experiments conducted in this study demonstrate that OAI-based DL, together with a P2P archiving system, can preserve information for a long period of time. The importance of the preservation of each digital object—measured in the model by the desired reliability—impacts on different lifetime of this object. Objects that were inserted with a high desired reliability had longer lifetime. Thus, we can conclude three main characteristics for our system:

6.1 OAI-PMH integration

It is intended for OAI-based systems to keep unchanged all OAI-PMH functionalities. This feature is important, as OAI-PMH is widely used and it can be considered a standard to build DL.

6.2 Independence of object preservation

Different information requires different storage time. Collections of photos, journals, and articles might need few years of storage; other information such as digital objects in museums and libraries requires hundreds of years. Our system allows flexibility in the choice of lifetime of objects to be preserved.

6.3 Optimization of the storage resources

Storage repositories may suffer many types of damages on their contents, so each repository has a different reliability. Allowing each storage repository with a parameter capable of measuring the independent probability of failure is the closest way to model real networks. This approach allows the flexibility in the time of preservation of each object and therefore different numbers of replicas for them, impacting on a better usage of network storage repositories.

Future studies include the implementation of a complete digital preservation system. To do this, the system should be concerned with the auditory of the replicas and other threats such as software obsolescence, not considered in this study. Furthermore, very little emphasis was given to the items retrieving. Possibly, we can use our model of digital archiving on systems such as LOCKSS and BRICKS, and evaluate their feasibility. Finally, when the complete digital preservation system is finished, we intent to implement a whole distributed digital library, based on this study, in a real-world environment.

References

1. arXiv.org e-Print archive. www.arxiv.org. Accessed 30 Jan 2012
2. Agosti, M., Hans Jörg, H., Türker, C.: Digital library architectures: peer-to-peer, grid, and service-orientation. In: Pre-proceedings of the sixth thematic workshop of the EU network of excellence DELOS, S. Margherita di Pula, Cagliari, Italy, 24–25 June, 2004. Edizioni Libreria Progetto, Padova (2004). Accessed 24 July 2010
3. Ahlborn, B: OAI-P2P: A peer-to-peer network for open archives. In: ICPPW '02: Proceedings of the 2002 international conference on parallel processing workshops, p. 462. IEEE Computer Society, Washington, DC (2002). Accessed 24 July 2010
4. Amrou, A., Maly, K., Zubair, M.: Freelib: peer-to-peer-based digital libraries. In: AINA '06: Proceedings of the 20th international conference on advanced information networking and applications, vol. 1 (AINA'06), pp. 9–14. IEEE Computer Society, Washington, DC (2006). Accessed 30 Jan 2012
5. Brian, C., Hector, G.: Creating trading networks of digital archives. In: JCDL '01: Proceedings of the 1st ACM/IEEE-CS joint conference on digital libraries. ACM, New York (2001). Accessed 30 Jan 2012
6. Brisco, T.: DNS Support for load balancing. www.dl.acm.org. April 1995. Accessed 30 Jan 2012
7. Dabek, F., Frans Kaashoek, M., Karger, D., Morris, R., Stoica, I.: Wide-area cooperative storage with CFS. In: Proceedings of the 18th ACM symposium on operating systems principles (SOSP '01). Chateau Lake Louise, Banff, Canada, October (2001)
8. Flávio Rufino Seára, E.: Uma arquitetura OAI para Preservação Digital utilizando redes Peer-to-Peer Estruturadas. Master's thesis, Federal University of Paraná (2008)
9. Garey, M.R., Johnson, D.S.: Computers and intractability: a guide to the theory of NP-completeness. W.H. Freeman and Company, New York (1979)
10. Ghodsi, A., Alima, L.O., Haridi, S.: Symmetric replication for structured peer-to-peer systems. In: Proceedings of DBISP2P, pp. 74–85 (2005)
11. Haeberlen, A., Mislove, A., Druschel, P. Glacier: highly durable, decentralized storage despite massive correlated failures. In: Proceedings of NSDI'05. USENIX Association, Berkeley, CA (2005)
12. Ktari, S., Zoubert, M., Hecker, A., Labiod, H.: Performance evaluation of replication strategies in DHTs under churn. In: Proceedings of MUM '07. ACM, New York (2007)
13. Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Wells, C., Zhao, B.: Oceanstore: an architecture for global-scale persistent storage. In: Proceedings of ASPLOS-IX. ACM, New York (2000)
14. Lagoze, C., Van de Sompel, H.: The open archives initiative: building a low-barrier interoperability framework. In: JCDL '01: Proceedings of the 1st ACM/IEEE-CS joint conference on digital libraries, pp. 54–62. ACM, New York (2001)
15. Liskov, B., Ghemawat, S., Gruber, R., Johnson, P., Shrira, L.: Replication in the harp file system. In: Proceedings of ACM SIGOPS, Pacific Grove, CA (1991)
16. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: Proceedings of SIGMETRICS '02. ACM, New York (2002)
17. Maniatis, P., Roussopoulos, M., Giuli, T., Rosenthal, D., Baker, M.: The LOCKSS peer-to-peer digital preservation system. ACM Trans. Comput. Syst. **23**, 2–50 (2005)
18. Martins, V., Pacitti, E., Valduriez, P.: Survey of data replication in P2P systems. Technical report, INRIA (2006)
19. Milojicic, D.S., Kalogeraki, V., Lukose, R., Nagarajan, K.: Peer-to-peer computing. Technical report, HP Labs, Bristol (2002)
20. Mitzenmacher, M., Upfal, E.: Probability and computing : randomized algorithms and probabilistic analysis. Cambridge University Press, Cambridge (2005)
21. Open access and institutional repositories with eprints. www.eprints.org
22. Open Archives Initiative. www.openarchives.org
23. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: Proceedings of the SIGCOMM '01. ACM, New York (2001)
24. Risse, T., Knezevic, P.: A self-organizing data store for large scale distributed infrastructures. In: ICDEW '05: Proceedings of the 21st international conference on data engineering workshops. IEEE Computer Society, Washington, DC (2005)
25. Rowstron, A., Druschel, P.: Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In: Proceedings of ACM SOSP'01. Banff, Canada (2001)
26. Rowstron, A., Druschel, P.: Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems. Lecture Notes in Computer Science (2001)
27. Shudo, K., Tanaka, Y., Sekiguchi, S.: Overlay weaver: an overlay construction toolkit. Comput. Commun. **31**, 402–412 (2008)
28. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of the SIGCOMM '01, pp. 149–160 (2001)
29. Tansley, R., Bass, M., Stuve, D., Branschofsky, M., Chudnov, D., McClellan, G., Smith, M.: The D space institutional digital repository system: current functionality. In: JCDL '03: Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries, pp. 87–97. IEEE Computer Society, Washington, DC (2003)
30. The Digital Object Identifier System. www.doi.org

31. The Handle System. www.handle.net
32. Vignatti, T., Bona, L.C.E., Vignatti, A.L., Sunye, M.S.: Long-term digital archiving based on selection of repositories over P2P networks. In: IEEE P2P'09: Ninth international conference on peer-to-peer computing (2009)
33. Virtua tls. www.vtls.com.
34. Weibel, S., Kunze, J., Lagoze, C., Wolf, M.: Dublin core metadata for resource discovery, The Internet Society (1998)
35. Winett, J.: Definition of a socket, May (1971)
36. Xu, Y.: A P2P based personal digital library for community. In: PDCAT '05: Proceedings of the sixth international conference on parallel and distributed computing applications and technologies, pp. 796–800. IEEE Computer Society, Washington, DC (2005)
37. Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowicz, J.D.: Tapestry: a resilient global-scale overlay for service deployment. IEEE J. Sel. Areas Commun. **22**(1) January (2004)