Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2005 Proceedings

Americas Conference on Information Systems (AMCIS)

2005

Statistical Basics of a Reliable World Wide Web Peer to Peer Storage System

Nick Gehrke
University of Goettingen, ngehrke@uni-goettingen.de

Lutz Seidenfaden
University of Goettingen, lseiden@uni-goettingen.de

Rainer Baule
University of Goettingen, rbaule@wiso.uni-goettingen.de

Statistical Basics of a Reliable World-Wide Peer-to-Peer Storage System

Nick Gehrke
Institute of information systems, Dep. 2
University of Goettingen
ngehrke@uni-goettingen.de

Lutz Seidenfaden
Institute of information systems, Dep. 2
University of Goettingen
lseiden@uni-goettingen.de

Rainer Baule
Institute for Finance an Banking
University of Goettingen
rbaule@wiso.uni-goettingen.de

ABSTRACT

Peer-to-peer networks are highly distributed and unreliable networks. Peers log in and off the network at their own needs without any overall plan. In the real peer-to-peer case there are no central nodes planning the resources of the network or having an overview about the state of the network. The paper on hand describes and mathematically analyzes a storage algorithm allowing information to be stored within the network without the originator of the information needs to stay online. Information is optimally "blurred" within the network meaning that the information is reconstructable with a high probability and a long time interval, but stored as least redundant as possible. The main focus is to analyze the mathematical and statistical properties of the presented peer-to-peer storage algorithm. Technical procedures are described at a high level and need further improvement. Thus, the paper on hand is primarily purely statistically peer-to-peer theory at this stage of research.

Keywords (Required)

Decentralized Systems, Error Correction Codes, Peer-to-Peer.

INTRODUCTION

Peer-to-peer (p2p) Systems are highly decentralized loosely coupled systems which are coordinated by special algorithms. Besides the high degree of decentralization, it is important that the network does not need manual administration. The network must be organized in an intelligent and self-organizing manner. However, the fact that the network depends on specific resources (i.e. peers) is problematic. During a peer's downtime, the resource is not accessible. In an optimal case, the p2p network should react like a diversified stock portfolio: The performance must be stable even if some stocks go for a dive. This paper will develop a coordination algorithm that follows this principle. To a certain degree, peers can shut down or go offline, without losing the information stored. This can only be achieved by redundancy. As we will show later, the redundancy can be kept to a minimum which then leads "to a cheap safeguard of the portfolio". The paper will analyze statistical dependencies between the ability to reconstruct stored information, the redundancy and its lifecycle in the network.

Dealing with a p2p storage system is not a selfsatisfiying task. It is rather a basic research approach for new applications. Which applications will evolve is hard to estimate. One possibility is that users can share goods without using a marketplace as an intermediary. Furthermore, it enables the decentralized commercial trading of contents. These applications certainly do not rely on a storage system, but it can be seen as a generic building block for them.

STRUCTURE OF THE ANALYSIS

The paper on hand is structured as follows. First at all we give an introduction in the required steps needed to construct a distributed p2p storage algorithm. These required steps are the following: First we need an algorithm dividing the information for storage in small pieces which can be stored decentralized within the network. For this purpose we use Reed-Solomon (RS) Codes. Second we need an algorithm for the coordination of files within the network. For this purpose the use of an overlay network like e.g. Chord or Pastry is necessary. These two required steps are explained briefly. After that we explain a p2p layer model and try to integrate our storage algorithm in this general layer model.

The succeeding sections describe our storage algorithm. Because this is mainly a paper conducting a statistical analysis of a p2p storage network, the description of the algorithm is at a very high level and needs more technical improvement.

The main part of the paper considers the statistical properties of the storage network. Especially the following issues are considered:

- The connection between redundancy and the probability to reconstruct a piece of information and the possibility to reduce the level of redundancy for a given reconstruction security, or, rather, probability.
- The calculation of the "lifespan" a piece of information has, and the information's lifespan for a given reconstruction security, or, rather, probability.

RELATED WORK

Research has been done in the field of distributed information storage. Several distributed storage applications have been described in the relevant literature, such as Oceanstore [Kubiatowicz, Bindel, Chen, Czerwinski, Eaton, Geels, Gummadi, Rhea, Weatherspoon, Weiner, Wells and Zhao 2000], Farsite [Adya, Bolosky, Castro, Chaiken, Cermak, Douceur, Howell, Lorch, Theimer and Wattenhofer 2000], PAST [Druschel and Rowstron 2001], Freenet [Clarke, Sandberg, Wiley and Hong 2000] or Freehaven [Dingledine, Freedman and Molnar 2000]. Like Oceanstore, we aim to build a storage system that can reconstruct information from an *untrusted infrastructure* and we also allow *nomadic data*, i.e. data can flow when refreshed by the system. Furthermore, like Oceanstore and PAST, the storage mechanism described in this paper also relies on a routing overlay network. We furthermore use error-correction codes like RS Codes (see 3.1) that can be used for secure storage in distributed systems [Plank 1997; Cooper and Garcia-Molina 2002], [Wheatherspoon and Kubiatowicz 2002]. But unlike e.g. Freehaven, the system described here does not rely on a community of servers (servnet). Any computer is able to join or leave the storage system.

Although we share some similarities with the systems mentioned above, we describe different mechanisms to store and reconstruct information in the system. However, this paper focuses on the statistical evaluation of a distributed storage system, not on its implementation – at least at this stage of research.

FUNDAMENTALS

Error Correction und Reed-Solomon Codes

RS Codes are used for error-correction. Two main categories of errors exist. On one hand, there are randomly distributed errors which affect single bits (random errors), on the other hand, there are clustered errors that affect hundreds or thousands of bytes (burst errors). RS Codes are particularly good at correcting the latter [Matthews 2001, p. 55]. The math behind the codes is not described in this paper. The interested reader is referred to the relevant literature [see e.g. Wade 2000, p. 277ff], [Wicker and Bhargava 1994, p. 1-8]. The property of RS Codes we need in the paper on hand is:

Let there be N storage devices (peers), D_1, D_2, …, D_N, each of which holds K bytes. These are called the "data devices". Let there be M more storage devices C_1, C_2, …, C_M, each of which also holds K bytes. These are called the "checksum devices". The contents of each checksum device will be calculated from the contents of the data devices. The goal is to define the calculation of each C_i such that if any M of the D_1, D_2, …, D_N, C_1, C_2, …, C_M fail, then the contents of the failed devices can be reconstructed from the non-failed devices (peers) [Plank 1997, p. 2].

Examples for the use of RS Codes include error-correction for compact discs [Immink 1994, pp. 43-58], communication systems [Wade 2000, p. 277], spread-spectrum communication systems (e.g. mobile phone networks) [see Wicker and Bhargava 1994, p. 11]. For a detailed discussion about the usage of RS Codes in those systems see [Pursley 1994; Sarwate 1994].

OVERLAY-STRUCTURES

Before we are able to explain the storage system, we have to shed light on the general topic of overlay networks, which will be used by the system.

While filesharing systems compete to fill the vacuum left by Napster (http://www.napster.com), research has put its focus on an alternative design for p2p networks. These networks are structured and therefore fundamentally different from earlier networks such as Gnutella (http://www.gnutella.com) which are organized in an unstructured manner. Structured networks are based on a certain logical graph structure e.g. a ring which guarantees a reliable and efficient search for content, i.e. objects will be found in logarithmic runtime. In order to build up a structured p2p network, two fundamental questions have to be answered [Peterson and Davie 2003]:

[1] How can objects (content) be mapped to nodes?

[2] How can requests be routed to the node responsible for a certain object?

In order to answer the first question, the well known hash table technique, which maps objects to addresses, is used. The result is a distributed hash table which is the backbone of several overlay protocols such as Chord [Stoica, Morris, Karger, Kaashoek and Balakrishnan 2001], [Dabek Brunskill, Kaashoek, Karger, Morris, Stoica and Balakrishnan 2001] Pastry

[Rowstron and Druschel 2001] and Tapestry [Zhao, Kubiatowicz and Joseph 2001]. In order to answer the second question, different routing strategies can be followed. The reader is referred to the relevant literature. For general security considerations in structured overlay networks see [Castro, Druschel, Ganesh, Rowstron and Wallach 2002] and [Sit and Morris 2002].

DISTRIBUTED STORAGE IN P2P NETWORKS

Requirements and Prerequisites

The requirements for a storage system are similar to those for other distributed systems: scalability, safety, load balancing, availability and robustness.

In this paper we will mainly focus on the safety and availability of information. In order to meet the aforementioned requirements, it is necessary to fulfill two main prerequisites. Every peer needs a pair of keys (consisting of a private and a public key) for signing, encrypting and checking the integrity of the information. In addition, peers need to have a certificate which ensures their identity. Both the pair of keys and the certificate have to be issued by an independent trust center. Although a central trust center may contrast with the idea of a genuine P2P-System we think that it is feasible to have such a trust center since it is only used once when a peer logs on for the very first time. The certificate is needed for the identification of peers when they access information and is vital for the confidentiality of information. The certificate also contains the public key of the peer. As a second prerequisite, an efficient distribution and retrieval mechanism for the information needs to be implemented.

Overlay-Layer-Model

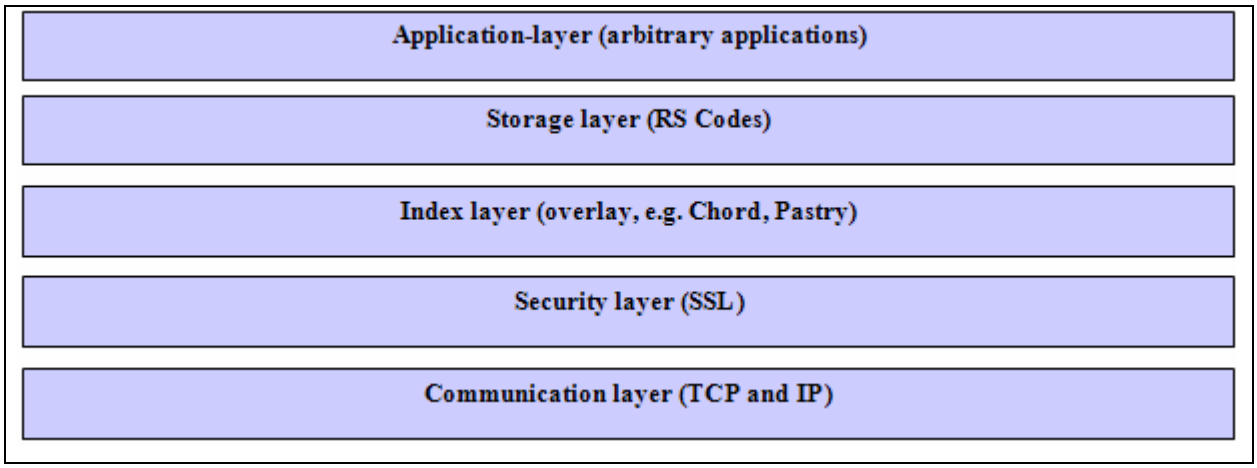


Figure 1: Overlay Layer Model

In the following, we will describe a layer model which is responsible for the realization of the proposed storage system's functions.

Communication layer

This layer fulfills communication tasks. It is already realized e.g. as the internet, which represents an overlay network based on telephone wires. TCP and IP is used as the communication protocol.

Security layer

This layer makes secure communication mechanisms (e.g. SSL) available for the communication layer. Furthermore, it contains the encryption and decryption functions for index and storage layers. It also manages the certificates and (public) keys of the peers.

Index layer

On this layer, the objects are indexed. In order to obtain a unique assignment of objects to storage peers, a unique fingerprint of each object is generated via hash functions. The allocation based on the fingerprint can then be realized by an arbitrary routing overlay (e.g. Chord, Pastry), which supports allocation in logarithmic runtime.

Storage layer

This layer handles the storage and retrieval of information. Subsequently, the storage process is described in more detail. Hereby, we need to differentiate between cases where a user stores information specifically for certain other users (private information) and cases where a user simply stores public information. The storage layer offers two fundamental methods for storing and retrieving information: one to put a file in the network and the other one to get it back. They both use functionalities of overlay networks for the unique mapping of information fragments to peers and the efficient retrieval of information fragments.

In order to ensure that data cannot be manipulated, we need to save the information fragments in such a way that the peers are unable to manipulate the information fragments they are in charge of. Peers should not be able to violate the information's integrity. In the following section, we will describe suitable encoding mechanisms.

Application layer

The application layer stores arbitrary applications which are dependent on a reliable and efficient storage. These applications take advantage of the different layers.

Saving information

After we have described the layer model, we will now describe how information can be saved in a decentralized network. The individual layers' functions are interconnected during the saving process.

[1] Encode the original information symmetrically with the help of a temporary symmetrical key.

[2] Determine the hash value of the encoded original information.

[3] Encode the temporary symmetrical key with an asymmetrical key and add this to the encoded original information. Use the following devices as an asymmetrical key:

  i. Your own private key if the information should be publicly accessible (public information).

  ii. The recipient's public key if the information should only be accessible by one recipient (private information).

[4] Divide the encoded original data in N parts and determine the M checksum parts.

[5] Determine the hash values of all N and M information fragments.

[6] Use your own private key to encode all hash values (= information fragment's signature) and add these fragments to the corresponding $N + M$ information fragments.

[7] Randomly select the $N + M$ peers and transfer each of the $N + M$ fragments to one of the $N + M$ peers.

[8] Draw up a list of information fragments for each of the $N + M$ peers and transfer this list with the information fragment. Each list contains:

  i. The hash value of the encoded original document from [2].

  ii. Q randomly chosen hash values from the $N + M$ information fragments from [5]. With regard to Q's size see below.

  The list thus has the following length (in bytes): $(Q + 1) \cdot$ length of one hash value (in case of SHA1 20 $(Q + 1)$ bytes).

Each of the $N + M$ peers needs to index its information fragments (delegation to the index layer):

- with regard to the information fragment it received,
- with regard to the original document as a whole.

The aforementioned encoding mechanisms prevent manipulation to a great extent. However, participants are still able to delete information fragments. They are thus able to make the retrieval of the information much harder (see below) or they can even make it useless altogether. The following section will analyze this particular problem.

Retrieving the information

After we have described how we can place the information into a network, while eliminating the danger of manipulation, we will now illustrate how this information can be retrieved and reassembled.

The following applies:

A list which is related to one information fragment and has been, together with the according information fragment, assigned to a peer, contains Q randomly chosen hash values. We will also assume that a list cannot save the same hash value twice. The list Q's lengths should be as short as possible in order to use as little memory as possible. However, if Q is too short, we run the risk that individual information fragments or a group of information fragments are not part of any list. From a graph-theoretical perspective, we would thus have an unconnected graph. We should thus try to find out in how far the existence of an unconnected graph is connected to the list Q's length. We can then choose a length that is big enough so that it is highly unlikely that we will end up with an unconnected graph. A formal analysis of this problem is possible and we used multinomial coefficients for the solution. But because this question is not of central interest for this paper we do not present these special details here.

Transmitting information fragments

When a peer X decides to leave the network, he has to make sure that he divides all of his information fragments onto the active peers. This is necessary since the information saved on X still needs to be retrievable. Peer X thus chooses one active peer for each information fragment and gives him the information fragment as well as the corresponding list with addresses from peers who have related information or checksum fragments. Peers who receive new information fragments need to index these fragments in such a way that they can be retrieved in the future, despite the fact that they are saved in a different place.

Refreshing information

In order to make sure that a piece of information is not forgotten over the course of time, peers should start a refresh or a complete redistribution every once in a while (see below for a statistical analysis). If peers do not perform the refreshing process after a certain amount of time, which could be interpreted as the information's expiration date, the number of information and checksum fragments needed for the reproduction of the information will decrease and thus the information will soon be forgotten.

In this context, all active peers need to determine how old their information is. If a piece of information has been saved for longer than T, the peer will start the process to retrieve the whole information (see above) and go through the saving process again. This means that the information will be re-divided into $N + M$ information fragments. This process will counteract the loss of information that might have happened already. During this process, the information will get new indexes as well.

STATISTICAL ANALYSIS OF THE PEER-TO-PEER MEMORY

Loss of information

After the last sections illustrated the technological possibility of a p2p memory, we will now describe the statistical characteristics of saving information in a p2p network. Through the usage of cryptographic algorithms, we can ensure that the information cannot be manipulated (see above). In a next step, we thus have to ask ourselves how we can guarantee a sufficient availability of the information. After all, peers who save the information can shut down at any time. This would result in a situation where the original information could not be reconstructed anymore. Based on the fundamental stochastic availability of peers, how can we deduce the probability that the original information is still available? Thus, the central question is:

How likely is the possibility that an information fragment is still available after a certain time T?

In order to establish this, we will need a stochastic model for the availability of peers in a network. It is reasonable to assume that the "log-off rate" $\lambda(t)$, which determines the probability $\lambda(t)\,dt$ that a peer logs off in an infinitesimal interval dt, is constant over time. This is consistent with a broad range of similar examples, e.g. the lifespan of electronic devices, the duration of telephone calls etc. As a consequence, the random length of the time interval in which a peer saves an information fragment (before the peer logs out again) is exponentially distributed. This implies that most peers are only available for a short period of time. Only a small number is present for a long time. The probability density function of the time intervals in which peers save their data is thus expressed in the following way:

$$(1) \qquad f_\lambda(t) = \lambda \cdot e^{-\lambda t}$$

with some parameter λ (the constant log-off rate). As mentioned before, when one peer logs off, he transfers his information fragment to a different peer. We will assume that this swap does not always take place correctly, but is actually prone to mistakes. For instance, a peer may crash, the electricity may be cut off, the network connection may be disconnected or a

peer neglects to swap intentionally when he logs off. A correct swap thus only takes place with the probability $v < 1$. We can interpret this fact as a kind of forgetfulness process. As we further assume that successive swaps are independent, according to the well known link between the exponential distribution and the Poisson process the number of swaps N_T within a given time T is Poisson-distributed with parameter λT:

$$(2) \qquad P(N_T = n) = \frac{(\lambda T)^{n-1}}{(n-1)!} e^{-\lambda T}$$

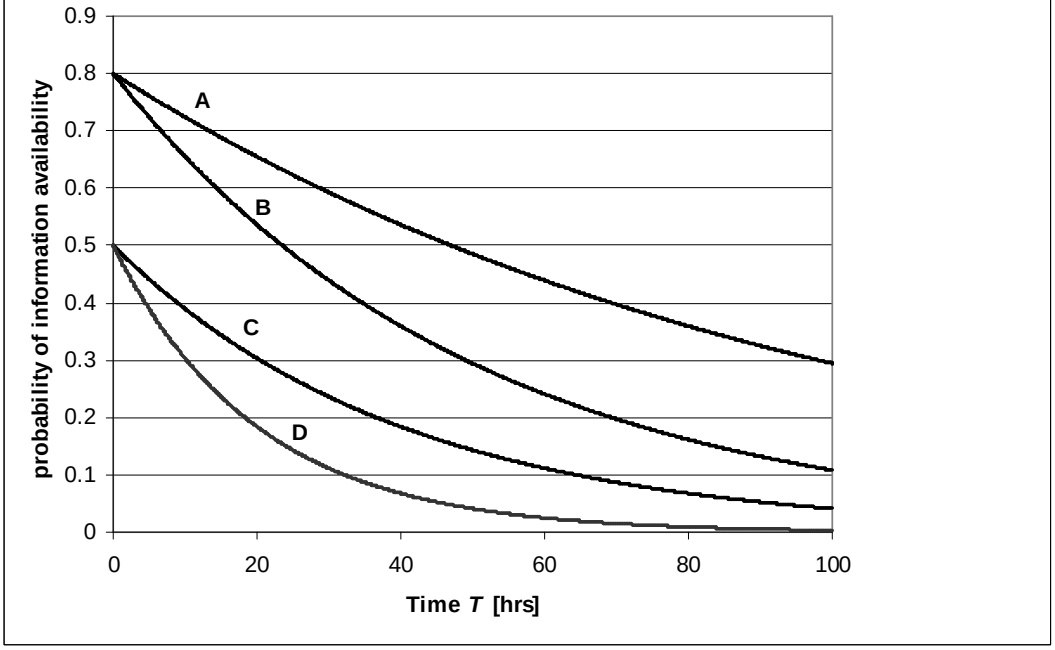
(2) determines the probability that exactly n swaps happen before T. However, this does not answer our initial question which asked for the probability that the information fragment is still available at point T. Due to the assumed independence, the probability that an information fragment "survives" at least n swaps (meaning that no mistake happens in the meantime) is simply v^n. With this information in hand, we calculate the probability that the random survival time τ of an information fragment is greater than a given time T as the sum of all probabilities conditional on the realization of the number of swaps N_T:

$$(3) \qquad P(\tau > T) = \sum_{n=1}^{\infty} P(\tau > T \mid N_T = n) P(N_T = n) = \sum_{n=1}^{\infty} \frac{(\lambda T)^{n-1}}{(n-1)!} e^{-\lambda \cdot T} \cdot v^n = e^{-\lambda \cdot T} \cdot v \cdot \sum_{n=0}^{\infty} \frac{(v \lambda T)^n}{n!}$$

Since the sum on the right hand side equals the power series expression of the exponential function, we get:

$$(4) \qquad \boxed{P(\tau > T) = v \cdot e^{(v-1)\lambda T}}$$

The following graphs express the availability of an information fragment, with four different parameter combinations, over the course of time:



	Scenario A	Scenario B	Scenario C	Scenario D
v (Probability that a transfer is conducted correctly)	0.8	0.8	0.5	0.5
λ (Distribution parameter savings interval)	0.05	0.1	0.05	0.1
Expected savings interval of a peer =1 / λ [hours]	20	10	20	10

Figure 2: Probability that an information fragment is still available over the course of time

The connection between forgetfulness and redundancy

(4) helps us to calculate the probability that an information fragment is still available at T. In the following, we will use this piece of information in order to determine how redundantly an information fragment has to be saved in a p2p network if we want to make sure that the probability that it can be reconstructed in the end is as high as an intended minimum value. In this

context, however, we are not simply interested in the availability of one information fragment, but in our ability to reconstruct the entire information. As we explained earlier in this paper, the original information was divided into N fragments and M additional checksum fragments with the help of, for instance, RS Codes. These fragments are then distributed to $N + M$ peers. The central question is thus:

How can we determine the ideal number of information fragments (N) and checksum fragments (M), if we want to make sure that the probability that the original information can be reconstructed – despite the fact that peers are unstable at times – is as high as possible, while the data redundancy remains relatively small?

This analysis requires the following parameters:

Parameter	Connotation
p	Probability that a peer is *not* available and that an information fragment cannot be accessed
N	Number of storages peers and fragments
M	Number of checksum peers and fragments
K	Number of bytes that a peer needs in order to save the information
D	Size of the original information (in bytes)
R	(Overall) data redundancy that is caused by the saving process (which is based on RS Codes)
W	Probability that the original information can be reconstructed with the help of storage and checksum peers (given N and M)
α	*Intended* probability that information can be reconstructed

Table 1: Parameter necessary to determine *N* and *M*

First of all, we have to determine the level of redundancy. Each of the $N + M$ peers – whether it is a storage or a checksum peer – saves exactly K bytes. Hence, a total of

$$K(N+M) = D\left(1+\frac{M}{N}\right) \tag{5}$$

bytes is distributed onto all peers. When it comes to redundancy, however, we are not interested in determining the absolute data redundancy, but we are interested in a relative redundancy value. We will therefore divide (5) by D. Redundancy R is thus defined as the relative excess saving capacity:

$$R = \frac{M}{N} \tag{6}$$

(6) represents one of the central tenets in our optimization problem. Redundancy R is intended to be as small as possible. However, we also need to account for our second central tenet: the demand that the probability that we are able to restore the original information (reconstruction probability W) is as high as possible. In the following, we will thus determine the reconstruction probability.

When are we able to reconstruct the original information? As we explained earlier in our description of RS Codes, we can only reconstruct the original information if fewer than M of the $N + M$ information fragments (in this case: peers) are unavailable. Each of the $N + M$ information fragments, or rather, peers, is available with a probability of $1 – p$ (which can be calculated with equation (4)). After all, the overlay structure – because it is based on the hash function which distributes information fragments in a uniform fashion – ensures that the peers are selected on a quasi-random basis. Therefore, it is plausible to assume that peers are stochastically independent. Hence the total number of available peers is binomially distributed with parameter $1 – p$. The reconstruction probability (i.e. the probability that not more than M of $N + M$ peers are unavailable) can thus be calculated in the following way:

$$W = \sum_{i=0}^{M} \binom{N+M}{i} \cdot p^i (1-p)^{N+M-i} \tag{7}$$

In order to conduct a formal analysis, we apply the de Moivre-Laplace limit theorem to approximate the binomial distribution with the continuous normal distribution. Mean and variance are given by $p\ (N + M)$ and $p\ (1 – p)\ (N + M)$, respectively, so it follows (using the correction term 0.5):

$$W \approx \Phi\left(\frac{M + 0.5 - p \cdot (N + M)}{\sqrt{p \cdot (1-p) \cdot (N + M)}}\right) \quad (8)$$

with Φ() distribution function of the standard normal distribution.

Through (6) and (8), the redundancy and the reconstruction probability are provided. Before we will analyze the optimization of variables *N* and *M*, we will present a rather intuitive graphic approach to the problem of optimization. Because *N* and *M* are both objects of the optimization, we face a two-dimensional optimization problem. Figure 3 displays the redundancy *R* and the reconstruction probability *W* in relation to *N* and *M*.

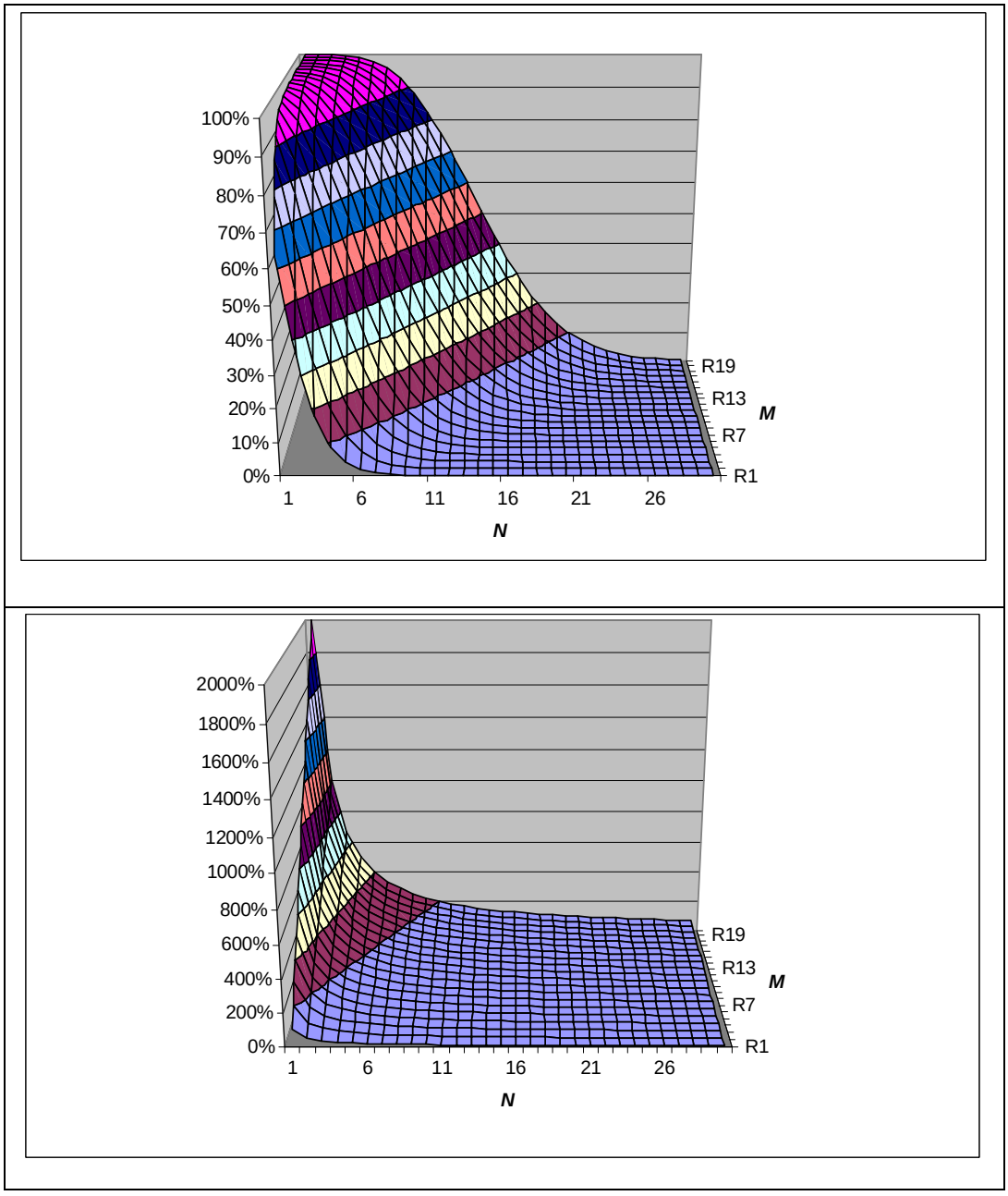


Figure 3: Reconstruction probability *W* (upper) and redundancy *R* (lower)

In the following, we will analyze this optimization problem in a formal manner. In order to achieve this goal, we will fix a specific level of reconstruction probability α. The required redundancy is the object of minimization. However, as there is a monotonic relationship between the reconstruction probability and the redundancy, this is not a real optimization problem. Rather, we can calculate the required number of checksum peers *M* to achieve the given reconstruction probability α. This is done by solving the equation

(9) $$\Phi\left(\frac{M+0.5-p\cdot(N+M)}{\sqrt{p\cdot(1-p)\cdot(N+M)}}\right)=\alpha$$

which yields

(10) $$M=\frac{0.5\,p\cdot\Phi^{-1}(\alpha)^2+p\cdot N-0.5+0.5\cdot\Phi^{-1}(\alpha)\sqrt{(4\cdot N-2+p\cdot\Phi^{-1}(\alpha)^2)\cdot p}}{1-p}.$$

For the redundancy R it follows:

(11) $$R(N)=\frac{p}{1-p}+\frac{p\cdot\Phi^{-1}(\alpha)^2-1+\Phi^{-1}(\alpha)\sqrt{(4\cdot N-2+p\cdot\Phi^{-1}(\alpha)^2)\cdot p}}{2(1-p)N}$$

By analyzing this equation, we can see that the redundancy converges towards a specific level with increasing N:

(12) $$R(N\to\infty)=\frac{p}{1-p}.$$

This level can be accepted as the smallest theoretical redundancy possible. If we substitute p with the result of the equation (4), and then take the connection with the survival time T into consideration, we have:

(13) $$R(N\to\infty,T)=\frac{1}{\nu}\cdot e^{\lambda T(1-\nu)}-1$$

Equation (13) shows that an asymptotic redundancy with advancing time T has to increase exponentially in order to guarantee a reconstruction probability α. Interestingly, the level of the asymptotic redundancy is independent of α. However, as seen in equation (11), the reconstruction probability influences the course of the redundancy for finite values of N. At this point, we would like to offer some examples for the course of the redundancy.

The following graph depicts the required redundancy (11) with different reconstruction probabilities α dependent on the number of information fragments N. In this example, the non-availability probability of an information fragment is $p = 0.20$. The asymptotic redundancy is therefore 0.25.

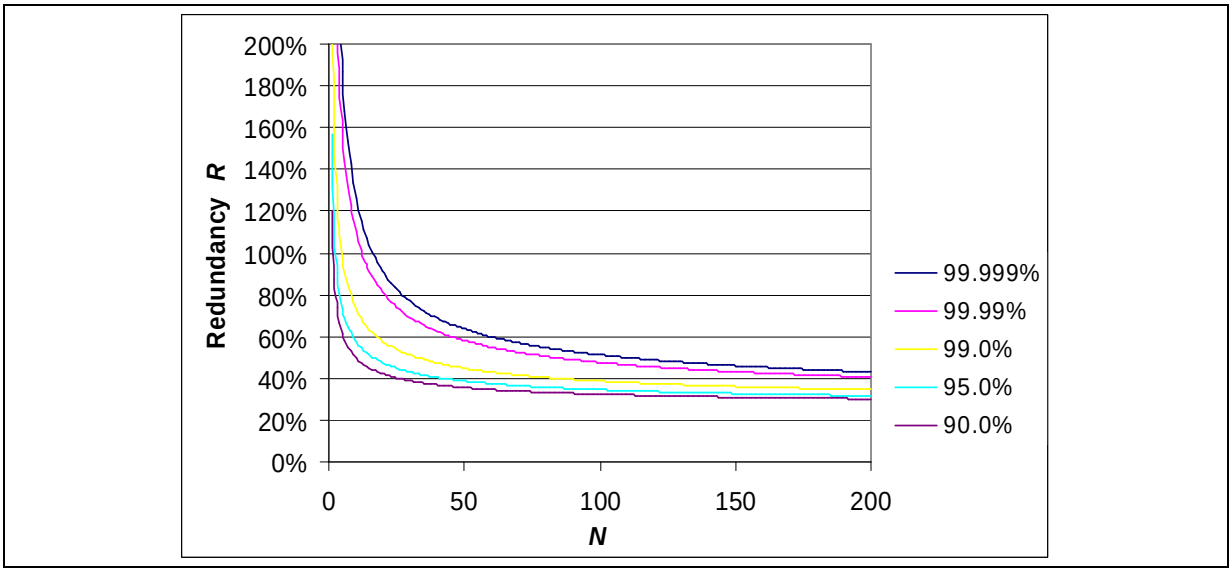


Figure 4: Required redundancy with differing reconstruction probabilities and growing number of information fragments *N*.

The graph shows that a large number of information fragments N results in a very low level of redundancy. However, a larger reconstruction probability does not increase the redundancy that much. Thus a high level of reconstruction probability does not drain resources.

The fight against forgetting

As explained, the information is refreshed after a specific time period T. The reconstruction probability after this time period is α. Hence, the probability that the information survives n time periods is α^n. The random lifespan L of the stored original information is geometrically distributed with $P(L = n\ T) = \alpha^n\ (1 - \alpha)$. The mean is given by:

(14) $$EL = \sum_{n=1}^{\infty} nT\alpha^n(1-\alpha) = T \cdot \frac{\alpha}{1-\alpha}$$

At this point, we are not only interested in the mean lifespan but also in a confidence level of the lifespan. For instance, we would calculate a guaranteed lifespan which is achieved with 99% probability. By applying the continuous geometric distribution with distribution function $F(t) = 1 - \alpha^t$, we get

(15) $$\boxed{L_{0.99} = T \cdot \left(\frac{\ln 0.99}{\ln \alpha} \right)}$$

The following numerical example illustrates this equation:

Given:	
T (time between two swaps)	50 hours
v (probability of transfer success)	0.8
λ (distribution parameter, average time of storage interval)	0.05 hours
Reconstruction probability α	0.99999
Expected store interval peers =1 / λ	20 hours
Offline probability of a peer in T	51.5%
Expected number of swaps	99999
Left 1% quantile of the geometric distribution	1005.03
Expected availability of information	570.8 years
99% guaranty of availability	5.736 years
Asymptotic redundancy	106.1%

Table 2: Calculation of availability of information

With all these numerical examples, the reader needs to take the following into consideration: We assume that the information fragments will disappear completely if one peer goes offline (with the probability p). Yet, practical experience shows that an information fragment that had initially disappeared will come into existence when one peer decides to go online again after some time. In this respect, the premises of the statistical model are chosen restrictively. The calculations need to be interpreted conservatively.

Parameter estimation in reality

In the previous sections, we presented the statistical characteristics of information storage in an unreliable p2p network. In order to have efficient information storage, peers need to pass and, after a certain point in time, reconstruct information fragments. For this process, peers need to know the global parameters λ and v. Yet, a central unit that collects these parameters is lacking. This problem can be solved as follows: Each peer takes a small sample of the parameter in question asking other peers. Having this small sample each peer asks again other peers randomly chosen for the mean of the samples taken before by the other peers. This can be done "several rounds" leading to a "cascading sample" with huge sample size without having too much effort. An analysis of such a "cascading sample" is straightforward and is not given here.

CONCLUSION

In this paper, we have demonstrated and analyzed an algorithm which allows us to securely store information in a decentralized p2p network. The following issues were analyzed in more detail:

- The correlation between redundancy and reconstruction probability and thus a potential minimization of redundancy with a given reconstruction security.
- The calculation of the information's lifespan, meaning the calculation of a guaranteed lifespan with a given reconstruction security or rather reconstruction probability.

The following problems are challenges that need to be further addressed:

Similar to the information fragments, the original information is indexed according to its hash value. One or several peers can be responsible for this task (if the index layer functions redundantly). If many peers request specific original information, the responsible index peers may be overloaded which results in a denial of service situation. It is problematic that the number of index peers is not scaled according to the demand. In order to achieve this, we would have to judge the demand to later retrieve this data (both the demanding peers and the index peers would have to be involved in this process).

Furthermore research on application needs to be conducted which would enable the usage of the new p2p storage possibilities. Options such as additional layers on top of the storage layer need to be explored. Database layers or transaction layers which would enable peers to complete transactions in a secure yet decentralized manner would serve as examples.

REFERENCES

1. Adya, W. J., Bolosky, M., Castro, R., Chaiken, G., Cermak, J. R., Douceur, J., Howell, J. R., Lorch, M., Theimer, R. P. and Wattenhofer, R. P. (2002) "FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment", 5th OSDI, 2002, http://research.microsoft.com/sn/Farsite/OSDI2002.pdf, 2002.
2. Castro, M., Druschel, P., Ganesh, A., Rowstron, A. and Wallach, D. (2002) Secure routing for structured peer-to-peer overlay networks, http://www.cs.rice.edu/~druschel/publications/security.pdf, 2002.
3. Clarke, I., Sandberg, O., Wiley, B. and Hong, T. (2000) Freenet: A Distributed Anonymous Information Storage and Retrieval System. In Proc. of the ICSI Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, 2000, pp. 311-320.
4. Cooper, B. and Garcia Molina, H. (2002) Peer-to-Peer Data Trading to Preserve Information, *ACM TOIS*, Vol. 20, No. 2, 2002, pp. 133-170.
5. Dabek F., Brunskill, E., Kaashoek, F., Karger, D., Morris, R., Stoica, I. and Balakrishnan, H. (2001a) Building Peer-to-Peer Systems With Chord, a Distributed Lookup Service, *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, 2001.
6. Dabek, F., Zhao, B., Druschel, P., Kubiatowicz, J. and Stoica,I. (2001b) Towards a Common API for Structured Peer-to-Peer Overlays, http://www.cs.berkeley.edu/~ravenben/publications/pdf/apis.pdf, 2001.
7. Dingledine, R., Freedman, M. and Molnar, D. (2000) The Free Haven Project: Distributed Anonymous Storage Service, URL: http://www.freehaven.net/doc/berk/freehaven-berk.ps, 2000.
8. Druschel, P. and Rowstron, A. (2001) PAST: A large-scale, persistent peer-to-peer storage utility, URL: http://research.microsoft.com/%7Eantr/PAST/hotos.pdf, 2001.
9. Immink, K. (1994) Reed-Solomon Codes and the Compact Disc, in Wicker, S., Bhargava, V. (Eds.) *Reed-Solomon Codes and Their Applications*, New York, 1994.
10. Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weiner, W., Wells, C. and Zhao, B. (2000) OceanStore: An Architecture for Global-Scale Persistent Storage, 2000.
11. Matthews, A. (2001) BCH Codes for Error Correction in Transmission of Binary Data,*The UMAP Journal*, Vol. 22, No. 2, 2001, 129-156.
12. Peterson, L. and Davie, B. (2003) Computer Networks: A Systems Approach, Amsterdam, 2003.
13. Plank, S. (1997) A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems, *Software – Practice & Experience*, Vol. 27, No. 9, 995-1012, September 1997.
14. Pursley, M. (1994) Reed-Solomon Codes in Frequency-Hop Communications, in Wicker, S., Bhargava, V. (Eds.) *Reed-Solomon Codes and Their Applications*, New York, 1994.
15. Rowstron, A. and Druschel, P. (2001) Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems, in *Proc. IFIP and ACM Middleware 2001*, Heidelberg, Germany, November 2001.
16. Sarwate, D. (1994) Reed-Solomon Codes and the Design of Sequences for Spread-Spectrum Multiple-Access Communications, in Wicker, S., Bhargava, V. (Eds.) *Reed-Solomon Codes and Their Applications*, New York, 1994.
17. Sit, E. and Morris, R. (2002) Security considerations for Peer-to-Peer distributed hash tables, http:// www.pdos.lcs.mit.edu/chord/papers/sec.pdf, 2002.
18. Stoica,I., Morris, R., Karger, D., Kaashoek, F. and Balakrishnan (2001) Chord: A scalable peer-to-peer lookup service for internet applications, in *Applications, Technologies, Architectures,and Protocols for Computer Communication Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications San Diego*, 2001, 149-160.
19. Wade, G. (2000): Coding techniques: an introduction to coding and error control, Basingstoke, 2000.
20. Weatherspoon, H. and Kubiatowicz, J. (2002) Erasure Coding vs. Replication: A Quantitative Comparison, *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS 2002)*.
21. Wicker, S. and Bhargava, V. (1994) Reed-Solomon Codes and Their Applications, New York, 1994.
22. Zhao, B., Kubiatowicz, J. and Joseph, A. (2001) Tapestry: A Infrastructure for Faul-tolerant Wide-area Location and Routing, http://www.cs.berkeley.edu/%7Eravenben/publications/CSD-01-1141.pdf, 2001.