Self-organized Data Redundancy Management for Peer-to-Peer Storage Systems

Yaser Houri, Manfred Jobmann, Thomas Fuhrmann

Computer Science Department
Technische Universität München
Munich, Germany
{houri,jobmann,fuhrmann}@in.tum.de

Abstract. In peer-to-peer storage systems, peers can freely join and leave the system at any time. Ensuring high data availability in such an environment is a challenging task. In this paper we analyze the costs of achieving data availability in fully decentralized peer-to-peer systems. We mainly address the problem of churn and what effect maintaining availability has on network bandwidth. We discuss two different redundancy techniques – replication and erasure coding – and consider their monitoring and repairing costs analytically. We calculate the bandwidth costs using basic costs equations and two different Markov reward models. One for centralized monitoring system and the other for distributed monitoring. We show a comparison of the numerical results accordingly. Depending on these results, we determine the best redundancy and maintenance strategy that corresponds to peer's failure probability.

1 Introduction

Since the advent of the peer-to-peer (P2P) paradigm, many P2P storage systems have been designed and implemented, for example DHash [1], OceanStore [2], TotalRecall [3], and pStore [4]. Their main objective is to reliably provide persistent storage on top of unreliable, yet collaborating peers.

Persistent storage implies the availability and durability of the stored data: *Availability* assures that data can be retrieved at any time. *Durability* means that once the data are stored, they are never lost. Durable data can be unavailable for a certain period of time. Thus availability implies durability, while durable data are not always available. Maintaining data durability is less expensive than maintaining availability in term of bandwidth and storage overhead. In this paper, we will concentrate on availability and its costs in a fully decentralized self-organizing environment.

Maintaining availability of the stored data is a challenging task since in P2P systems, peers join and leave the network dynamically. Even worse, peers often leave the network ungracefully, e.g., due to link outage, disk failure, or unexpected user behavior. In any such case it is hard to determine whether the departure of a peer is temporary or permanent. If in case of a temporary outage the redundancy is replaced too early, the system unnecessarily consumes bandwidth. On the other hand, delaying the redundancy repair too long can put the resilience of the system at risk.

In this paper, we analytically discuss the cost of maintaining data availability in a DHT based storage system in terms of network bandwidth usage. We compare two architectures using two Markov reward models. We discuss the bandwidth cost only, because this is the bottleneck in today's systems.

The remainder of this paper is organized as follows: In section 2 we summarize the relevant related work. In section 3 we describe how we model the various redundancy systems for our analysis. Section 4 contains our analysis. In section 5 we show the numerical results of our analysis. Section 6 concludes with an outlook to future work.

2 Related Work

In recent years, many studies and implemented prototypes discussed the issues of data redundancy strategies, fragments placement and redundancy maintenance, both analytically and through simulation. They all argue about which redundancy strategy is better: replication or erasure codes, but to the best of our knowledge none of them addresses the issue of under which conditions and with which parameters it is better to use replication or erasure codes. In this paper, we will address this issue.

As we have mentioned above, many storage systems prototypes were implemented with different redundancy and maintenance strategies. DHash [1] uses replication to ensure data availability. DHash places the data on selected peers and uses eager repairing policy to maintain redundancy. TotalRecall [3] and OceanStore [2] both use erasure codes to reduce storage overhead. They place the data randomly on the participating peers. Unlike OceanStore, TotalRecall uses a lazy maintenance policy which allows the reintegration of the temporarily unavailable fragments and therefore reduces the maintenance bandwidth. Only the temporarily unavailable fragments that return before starting the maintenance process are reintegrated. Carbonite [5] extends this policy to allow full reintegration of the temporarily unavailable fragments. Carbonite uses a specific multicast mechanism to monitor the fragments availability.

Besides the mentioned prototypes, many analytical studies explore what redundancy configuration achieves a desired level of reliability. Many of them compare replication strategies to erasure code strategies. We believe that all of them neglect important aspects that we intend to cover with this paper.

Tati et al. [6] gave a simple analysis on how the temporary and permanent failures affect the maintenance overhead. They also studied how the fragment placement schemes affect the system capacity. But they addressed the redundancy maintenance only from the storage overhead perspective. Lin et al. [7] also compared replication and erasure codes in terms of storage overhead. They showed that replication strategy is more suitable in a low peer availability environment.

In [1, 3, 8] the authors argued that erasure code reach the same level of availability as simple replication while using much less storage space. Rodrigues et al. [9] argued back that erasure coding has its limitations, when taking the network bandwidth into account. They show that sustaining a high data availability level using erasure coding generates a great bandwidth overhead due to churn: When a peer leaves the network and another one joins, ideally the new peer would take over the data fragments that have just been lost. But in order to regenerate a data fragment, the whole object that it belongs to needs to be reconstructed first. Therefore, a new peer has to download all the fragments that are needed to reconstruct the objects for which fragments have been lost. This consumes a large amount of bandwidth, which is a limiting factor for the scalability of P2P storage systems [10]. Their solution is to maintain a copy of the whole object at one of the peers, while replicating fragments of the stored objects to other peers. Such a hybrid solution creates a bottleneck, because a peer has to replace all its objects' fragments when they get lost in the system. Vice versa the other peers have to ensure that peer's availability. Such a hybrid strategy adds great complexity because the system needs to maintain two types of redundancy.

Motivated by network coding [11, 12], Dimakis et al. developed a solution to overcome the complexity of the hybrid strategy [13]. They applied the random linear network coding approach of Ho et al. [12] to a *Maximum-Distance Separable* (MDS) erasure code for storing data. But unless the new peers keep all the data they download, the performance proved to be worse than erasure codes.

Acedaski et al. showed mathematically and through simulation [14] that random linear coding strategy achieves a higher availability level than erasure codes while consuming much less storage space. They suggest to cut a file into m pieces and store k random linear combinations of these pieces with their associated vectors in each peer. A similar strategy was developed by Gkantsidis et al. in [15] for content distribution networks. But in both papers, the authors neglected the cost for repairing the redundancy. Therefore their results come in favor of random linear coding.

Duminuco et al. [16] showed that using linear coding causes high costs for redundancy maintenance compared to simple replication. They introduced *hierarchical codes*, a new coding scheme which offers a trade-off between storage efficiency and repairing costs. But their scheme has higher repairing costs than replication.

Wu et al. [17] studied the availability of a data object analytically and through simulation. They developed their model based on stochastic differential equations. They analyzed the performance of the distributed storage system in terms of effective storage capacity and maintenance bandwidth considering proactive and reactive maintenance strategies. But in their analysis, the authors neglected the monitoring costs. Alouf et al. [18] did a similar analysis. But their concern was availability in terms of redundancy level, not costs.

Part of our analysis is similar to the one used by Ramabhadran et al. [19], but we calculate the bandwidth costs differently. Moreover, our concern, in contrast to Ramabhadran et al. [19], is the comparison between replication and erasure coding in terms of bandwidth costs.

Other studies, e.g., Datta et al. [20], did the comparison between replication and erasure coding not in terms of bandwidth usage, but in terms of durability using different repairing strategies and resilience to correlated failures. This is not in the scope of our paper.

The main problem is that increasing the amount of replication also increases the probability that one of the involved peers fails. The probability of losing a fragment increases proportionally to the number of peers that are involved in storing the fragments. For each peer that fails, we need to repair the redundancy and thereby consume network bandwidth. The more redundancy we have, the higher are the monitoring, maintenance, and storage costs.

In this paper, we investigate different redundancy and maintenance strategies and relate their bandwidth cost to the achieved availability. We consider this the single most important criterion for selecting a redundancy strategy, because today bandwidth costs dominate over storage costs: A typical DSL up-link provides up to 1 MBit/s, whereas a typical home user HD drive provides up to 1000 GB. To the best of our knowledge that relation has not been addressed analytically, so far.

3 System Model

We consider a P2P storage system where the peers randomly join and leave the system. When a peer disconnects, all the data that has been stored on that peer is no longer available. Therefore, a storage system must create redundancy in order to better ensure data availability.

In this paper we compare two redundancy mechanisms: replication and erasure coding. When applying the *replication* strategy, each block of data is replicated so that we altogether have n copies of that data block. Using the *erasure coding* strategy means to create n fragments of data, m of which suffice to reconstruct the data block. For the purpose of the paper we assume that all replica or fragments are distributed so that their potential loss is independent and identically distributed (i.i.d.). Furthermore, we attribute all loss events to ungracefully disconnecting peers that may or may not return. Concerning our analysis all other potential reasons are indistinguishable from these two reasons.

We have two possibilities of how to respond to disconnecting peers: In the first case, we consider a replica or fragment permanently lost, when the respective peer disconnects. Therefore, we immediately invoke the repair process (*eager repair strategy*). When the peer rejoins the network later, there is no point in reintegrating the recovered data again, because it has already been replaced. In the second case, we wait some time before we consider the data as lost and accordingly delay the repair process (*lazy repair strategy*). Therefore, when the peer rejoins early enough, we can omit the repair process and save the according costs.

Even though both strategies are different from a system perspective, they map to the same analytical model: Both cases can be described by a probing rate, a repair rate, the probing costs, and the repair costs. The probing rate or a repair rate can be related by the rejoining probability: The entire cost are the sum of all probes and the repair cost, when the strategy decides that the data has been permanently lost. (See analysis below.)

In order to manage the appropriate level of redundancy the system has to check which peers have disconnected. We assume a heartbeat model, where the peers mutually probe the existence of the other peers. In practice these probes must ensure that the peers actually take the effort of storing the data. But for the purpose of this paper, it is out of scope to discuss strategies that cope with malicious peers, because regardless of their actual workings all probing mechanisms map to a certain bandwidth effort. We only consider that effort.

We study two heartbeat models: In *model 1* a peer monitors one other peer at a time. If that peer is considered lost, the monitoring peer recreates one replica or redundancy fragment. In *model 2* a peer simultaneously monitors all other peers that store a replica or fragment of the respective block, so that it can recreate the redundancy level in one go, even if multiple replica or fragments have been lost.

4 Analysis

In this section we present a mathematical analysis of the data redundancy management models that we have introduced in section 3. As described already, we study only the effect of churn in term of bandwidth consumption, because all peer failures can be mapped to peer outages, and bandwidth cost are the dominant cost component today. The goal of our analysis is to find the redundancy strategies that fits best the different application scenarios.

4.1 Redundancy Repairing Costs

For our redundancy cost discussion we need to first define the peers' failure model. Ramabhadran et al. [19] found that peer failures can be modeled with an exponential distribution, i.e. we assume a failure model with a rate λ. Without loss of generality, we simplify and assume that each peer in our system stores only one unique block or its replica. From that we calculate the cost for monitoring and restoring the desired level of redundancy. In our equations we use the superscript "r" for replication and the superscript "e" for erasure codes.

Replication Redundancy Repairing Costs As peers fail, replicas are lost and need to be repaired. The resulting costs for the entire process are the sum of the monitoring costs (for $t >> \lambda^{-1}$) and the repairing costs for the failed replicas.

$$RC^r(t) = \lambda \cdot t \cdot size(b) + M(t) \tag{1}$$

λt is the number of failed peers at time t.

Even though the monitoring costs depend on the used monitoring protocol, they can be modeled by a simple *heartbeat protocol*.

$$M(t) = \mu \cdot t \cdot N \cdot size(heartbeat) \tag{2}$$

N is the number of peers that store replica or fragments in the system. In the case of replication, N is also the number of replica including the original data block. In the case of erasure codes, N corresponds to the number of fragments. $size(heartbeat)$ denotes the sum of all messages required for one heartbeat, and μ is the average heartbeat rate.

The monitoring costs are proportional to the redundancy level R_L. For replication, the redundancy level equals the number of desired replicas in the system. For erasure codes, the redundancy level is calculated as follows:

$$R_L = \frac{n}{m} \ , \tag{3}$$

where n is the total number of fragments including the redundant ones, and m is the required fragments to reconstruct the data object.

Erasure Codes Redundancy Repairing Costs For erasure codes, the repairing costs consists of the sum of all messages that are needed to reconstruct a lost fragment, and again the monitoring costs.

$$RC^e(t) = \lambda \cdot t \cdot m \cdot size(f) + M(t) \tag{4}$$

f denotes a the fragment and m denote the number of needed fragments to reconstruct the data block in order to regenerate the lost fragments.

It seems that both equation 1 and equation 4 deliver the same results, since $size(b) \simeq m \cdot size(f)$,but this is not the case. The costs for erasure codes are slightly higher due to the higher monitoring costs. This is because to reach the same replication redundancy level using erasure codes, we will need more peers than replication.

4.2 Markov Reward Model

In this section, we analyze the models from section 3 with the help of two different Markov chain models. From them we calculate the repairing costs using reward functions.

Our system consists of peers which store replicas or erasure code fragments. Each peer joins the system for some limited duration t_{on}. If a peer that recently left rejoins before the next probe discovers its absence, we treat the peer as if it never left the system. The same principle applies when the probing strategy is such that it waits for rejoining peers.

Measurements have shown [19] that t_{on} can be modeled as an exponentially distributed random variable with mean $\frac{1}{\lambda}$, where λ is the departure rate of the peers. We assume that t_{on} is independent and identically distributed for all peers in our system.

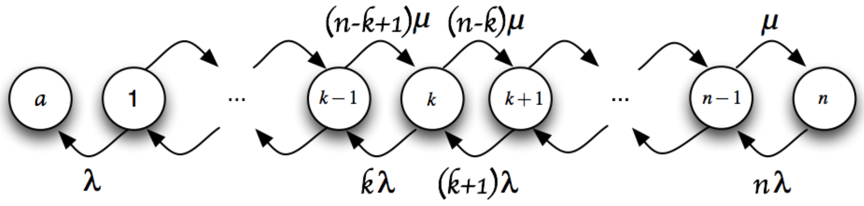
A data block b on a peer is available as long as the peer participates in the system. When a peer leaves, the data block b is considered lost. The system's repair mechanism compensates for this attrition and replenishes the lost replicas. To this end, it must first detect the loss of a replica or fragment, and then create a new one. In the case of erasure coding the peer must also download enough fragments before it can create the new fragment.

The whole repair process thus takes the system some time. We assume that this time, t_{rep}, is also an exponentially distributed random variable with mean $\frac{1}{\mu}$, where μ is the repair rate. That leads us to describe our system as a Markov chain.

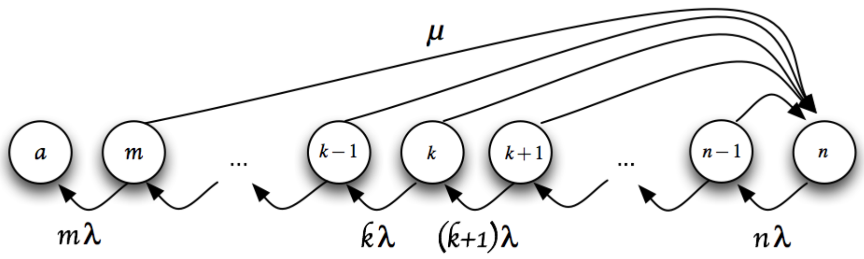
At some point of time, the system has k available replicas $(0 \leq k \leq n)$; the remaining $n-k$ are being repaired. In state k, any one of the k available replicas can fail. In such a case, the system goes to the state $k-1$. When one of the $n-k$ unavailable replicas is repaired, the system goes to the state $k+1$. The system moves from the state k to the state $k-1$ with rate $k\lambda$ and for the state k to state $k+1$ with rate $(n-k)\mu$.

For replication, the state 0 is an absorbing state. In the case of erasure codes, the state $m-1$ is an absorbing state. In the absorbing state, the system can no longer recover the original data block. The average time span between the system initialization and this event is the expected lifetime for that block.

Figure 1 shows Markov models for our two monitoring models.



(a) Model 1: Monitor one replica or fragment only



(b) Model 2: Monitor all replica or fragments

Fig. 1. Markov Chain Models

In this paper we compare both models in terms of bandwidth costs and mean life time. To this end, we add reward functions to our model.

In state i the system occupies an amount RR_{ii} of storage space. This is called the *rate reward*. In our case we find

$$RR^r_{ii} = i \cdot size(b) \tag{5}$$
$$RR^e_{ii} = i \cdot size(f) \tag{6}$$

Each transition from state i to state j consumes an amount IR_{ij} of bandwidth. This is called the *impulse reward*. In the case of replication the number of repaired replicas and the block size drive the impulse reward. In the erasure code case, it is driven by the number of fragments needed for reconstruction and the newly created fragments.

$$IR^r_{ij} = (j-i)size(b) \text{ , where } j > i \tag{7}$$
$$IR^e_{ij} = (m-1)size(f) + (j-i)size(f) \text{ , where } j > i \tag{8}$$

Since our comparison is in term of bandwidth costs, we only take the impulse rewards into account.

The impulse reward during the time interval $[t, t + \Delta t]$ is calculated as follows, see for example [21]:

$$IR_{ij}(t, t + \Delta t) := \overline{P}_i(t, t + \Delta t) \cdot \Delta t \cdot IR_{ij} \cdot \mu_{ij} \tag{9}$$

μ_{ij} is the mean repair rate for a transition from state i to state j, where $j > i$.

$$\overline{P}_i(t, t + \Delta t) := \frac{1}{\Delta t} \int_{\tau=t}^{t+\Delta t} P_i(\tau) \delta\tau \tag{10}$$

$P_i(t)$ is the probability of being in state i at time t. $\overline{P}_i(t, t + \Delta t)$ is the average probability of being in state i during the time interval $[t, t + \Delta t]$.

The impulse reward for all states at time interval $[t, t + \Delta t]$ is simply the sum over all states:

$$IR(t, t + \Delta t) := \sum_{i \in \mathbb{S}} \sum_{j \in \mathbb{S}} IR_{ij}(t, t + \Delta t) \tag{11}$$

To calculate the average total reward (i.e. bandwidth costs), we need to calculate the mean life time, T_l:

$$E[T_l] = -P_0 \cdot (Q_c)^{-1} \cdot \mathbf{1} \tag{12}$$

P_0 is the initial probability vector excluding the absorbing state. Q_c is the transition matrix Q excluding the absorbing state. $\mathbf{1}$ is a vector of ones of appropriate size.

The total expected repairing costs during time interval $[t, t + \Delta t]$ is then calculated by adding the total expected impulse rewards to the monitoring costs from equation 2 during that time interval:

$$RC(t, t + \Delta t) = IR(t, t + \Delta t) + M(t, t + \Delta t) \tag{13}$$

In our numerical evaluation, $t = 0$ and $\Delta t = E[T_l]$.

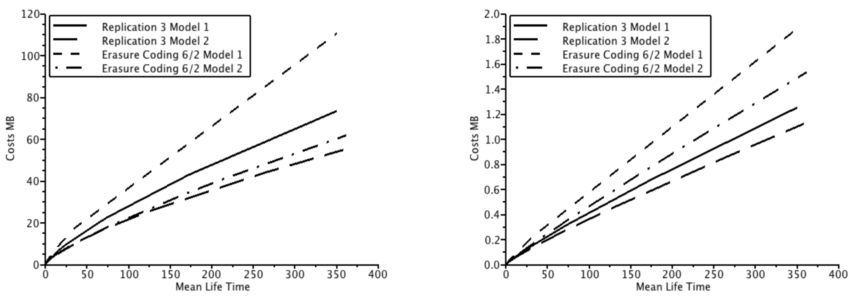
5 Numerical Results

In this section we evaluate our developed models with the following parameters:

- Block size: 512 KB and 4 KB
- Heartbeat messages (total volume): 512 Byte
- Replication redundancy level: $[3, 6]$
- Erasure codes with $n = [3, 14]$ and $m = [2, 10]$

In the calculations we use μ and λ values from company context measurement. This means that the presented life times are on the order of hundreds of days.

We evaluated our models according with different $\rho = \frac{\mu}{\lambda}$. Figure 2(a) shows the bandwidth costs difference between replication and erasure codes for block size 512 KB. In *model 1*, both replication 3 and erasure codes 6/2 reach the same mean life time for the same ρ, but the bandwidth costs for erasure codes are higher. The reason behind this difference is that erasure codes use more peers to store the fragments. The more peers, the larger is the probability that one of them disconnects in a given time interval.



(a) 512 KB data blocks (b) 4 KB data blocks

Fig. 2. Replication vs. Erasure codes

Thus, erasure codes need more repairing than replication and therefore consume more bandwidth.

Erasure codes need a lower monitoring frequency than replication to reach the same lifetime. Nevertheless, in model 1 erasure codes consume slightly more bandwidth than replication, because the replicating peer has to download several other fragments. In model 2 we repair all lost blocks or fragments at once. Here, erasure codes come cheaper than replication (in model 1), because for each erasure code repair, the required fragments need to be downloaded only once, and the size of a fragment is smaller than the size of a block.

When using a block size of 4 KB, erasure codes lose their advantages (cf. figure 2(b)). Now they consume considerably more bandwidth than replication in both models. This is due to the higher repairing rate and the higher monitoring costs.

From figures 3(a), 3(b), and 3(e) we can see that the bandwidth costs increase with the redundancy level.

From figures 3(f) we can deduce that increasing the number m of required fragments for erasure code scheme increases the bandwidth costs, even for the same redundancy level.

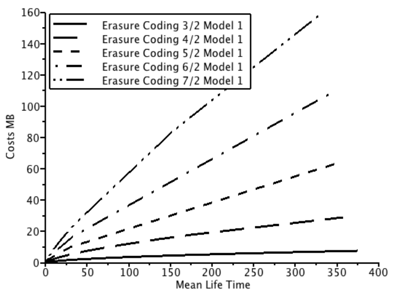
In figures 3(d) we notice that even if we decrease the redundancy level, the increase of the number m of required fragments for erasure codes increases the bandwidth costs.

The above results are also valid when using smaller block sizes (not shown here).

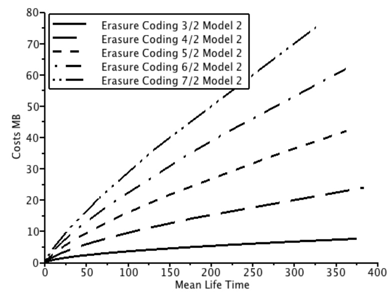
6 Conclusions and Future Work

In this paper we have analyzed the influence of the various parameters on the bandwidth costs of the respective redundancy strategies. In *model 2*, erasure codes prove itself as good as replication in terms of bandwidth consumption. But erasure codes requires a higher repair rate to achieve the same mean lifetime as replication,.

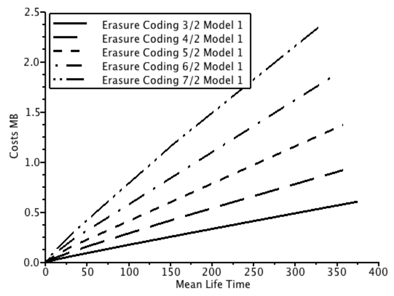
In *model 1*, replication outperforms erasure codes, because erasure codes involve more peers to store the fragments. Therefore the probability that a peer fails and a fragment gets lost is larger than in the replication case. Thus erasure codes need to



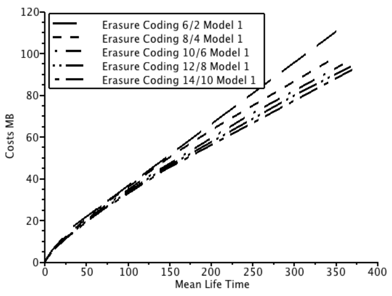
(a) Different erasure codes schemes with the same m using model 1 and 512 KB data blocks



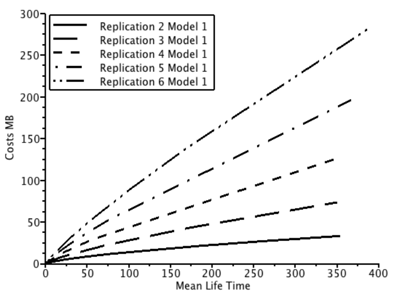
(b) Different erasure codes schemes with the same m using model 2 and 512 KB data blocks



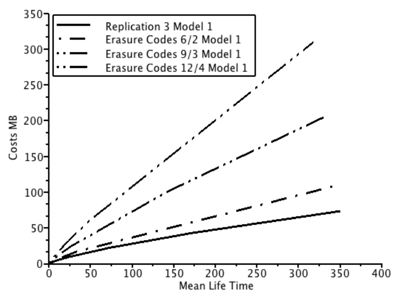
(c) Different erasure codes schemes with the same m using model 1 and 4 KB data blocks



(d) Different erasure codes schemes with constant $n - m$ using model 1 and 512 KB data blocks



(e) Different replication levels using model 1 and 512 KB data blocks



(f) Replication vs. different erasure codes schemes with the same redundancy level, using model 1 and 512 KB data blocks

Fig. 3. Effect of varying the parameters

repair the redundancy more often. Even worse, for each lost fragment, a peer needs to download m required fragments to generate the lost one.

In general, the repair costs increase with the redundancy level. The higher the redundancy level, the more peers are needed to store the blocks or fragments. The more peers involved, the higher the probability that one of them fails and the larger the effort to replace the lost replica or fragment. Thus, an increased monitoring frequency outperforms an increase level of redundancy.

The higher the ratio of required fragments to the total number of fragments, the smaller the repair costs, because the more required fragments we have, the smaller the fragment size and therefore the smaller the consumed bandwidth. The higher the repairing rate, the less replica or fragments we need to reach the desired average lifetime of the data.

Depending on the failure rate and the number of allowed simultaneous peer failures in our network, we can tune the redundancy level, the repairing rate and the redundancy scheme to achieve the required availability in the system.

There are some other techniques like *network coding* [13] and *hierarchical codes* [16]. These techniques trade bandwidth for storage space. As describe above, we consider this a bad trade, because today the bandwith costs dominate over the storage costs. Even more, the available bandwidth of the individual peers in the system determines the rate of the repair process. Therefore, the value of the repair rate is limited by the available bandwidth at the peers. It is among our future work to study the effect of the available bandwidth on the repair process in terms of delay. We also intended to study the effects of read requests and correlated failures on the repairing process.

Finally we note that this analytical work is part of the effort to improve IgorFS. In the course of these workings, we will implement a data redundancy management protocol in IgorFs [22] and evaluate it in PlanetLab.

References

1. Dabek, F., Li, J., Sit, E., Robertson, J., Kaashoek, M.F., Morris, R.: Designing a DHT for low latency and high throughput. In: in NSDI. (2004) 85–98
2. Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B.: OceanStore: An architecture for global-scale persistent storage. In: Proceeedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), Boston, MA (2000) 190–201
3. Bhagwan, R., Tati, K., chung Cheng, Y., Savage, S., Voelker, G.M.: Total recall: System support for automated availability management. In: In Proc. of NSDI. (2004) 337–350
4. Batten, C., Barr, K., Saraf, A., Trepetin, S.: pStore: A secure peer-to-peer backup system. Technical Memo MIT-LCS-TM-632, Massachusetts Institute of Technology Laboratory for Computer Science (2002)
5. Chun, B., Dabek, F., Haeberlen, A., Sit, E., Weatherspoon, H., Kaashoek, M., Kubiatowicz, J., Morris, R.: Efficient replica maintenance for distributed storage systems (2006)
6. Tati, K., Voelker, G.: On object maintenance in peer-to-peer systems (2006)
7. Lin, W.K., Chiu, D.M., Lee, Y.B.: Erasure code replication revisited. In: Fourth International Conference on Peer-to-Peer Computing (P2P'04), 2004. (2004) 90–97

8. Weatherspoon, H., Kubiatowicz, J.: Erasure Coding vs. Replication: A Quantitative Comparison. In: IPTPS. (2002)
9. Rodrigues, R., Liskov, B.: High availability in dhts: Erasure coding vs. replication. In: In Proc. of the 4th International Workshop on Peer-to-Peer Systems. (2005)
10. Blake, C., Rodrigues, R.: High availability, scalable storage, dynamic peer networks: pick two. In: HOTOS'03: Proceedings of the 9th conference on Hot Topics in Operating Systems, Berkeley, CA, USA, USENIX Association (2003) 1–1
11. Ahlswede, R., Cai, N., yen Robert Li, S., Yeung, R.W., Member, S., Member, S.: Network information flow. IEEE Transactions on Information Theory **46** (2000) 1204–1216
12. Ho, T., Mdard, M., Koetter, R., Karger, D.R., Member, A., Effros, M., Member, S., Member, S., Member, S., Shi, J., Leong, B.: A random linear network coding approach to multicast. IEEE Trans. Inform. Theory **52** (2006) 4413–4430
13. Dimakis, R.G., Godfrey, P.B., Wainwright, M.J., Ramch, K.: Network coding for distributed storage systems. In: In Proc. of IEEE INFOCOM. (2007)
14. Acedański, S., Deb, S., Mdard, M., Koetter, R.: How good is random linear coding based distributed networked storage. In: In NetCod. (2005)
15. Gkantsidis, C., Rodriguez, P.: Network coding for large scale content distribution. INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE **4** (2005) 2235–2245 vol. 4
16. Duminuco, A., Biersack, E.: Hierarchical codes: How to make erasure codes attractive for peer-to-peer storage systems. In: Eighth International Conference on Peer-to-Peer Computing, 2008. P2P '08. (2008) 89–98
17. Wu, D., Tian, Y., Ng, K.W., Datta, A.: Stochastic analysis of the interplay between object maintenance and churn. Comput. Commun. **31**(2) (2008) 220–239
18. Alouf, S., Dandoush, A., Nain, P.: Performance Analysis of Peer-to-Peer Storage Systems. Research Report RR-6044, INRIA (2006)
19. Ramabhadran, S., Pasquale, J.: Analysis of long-running replicated systems. In: INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings. (2006) 1–9
20. Datta, A., Aberer, K.: Internet-scale storage systems under churn – a study of the steady-state using markov models. In: P2P '06: Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing, Washington, DC, USA, IEEE Computer Society (2006) 133–144
21. Lampka, K.: A symbolic approach to the state graph based analysis of high-level Markov Reward Models. PhD thesis, University Erlangen-Nuremberg (2007)
22. Kutzner, K., Fuhrmann, T.: The IGOR file system for efficient data distribution in the GRID. In: Proc. Cracow Grid Workshop CGW 2006. (2006)