An Analytical Estimation of Durability in DHTs

Fabio Picconi[1], Bruno Baynat[2], and Pierre Sens[3]

[1] University of Bologna, Italy
picconi@cs.unibo.it
[2] LIP6, Paris, France
Bruno.Baynat@lip6.fr
[3] INRIA Rocquencourt, France
Pierre.Sens@inria.fr

Abstract. Recent work has shown that the durability of large-scale storage systems such as DHTs can be predicted using a Markov chain model. However, accurate predictions are only possible if the model parameters are also estimated accurately. We show that the Markov chain rates proposed by other authors do not consider several aspects of the system's behavior, and produce unrealistic predictions. We present a new analytical expression for the chain rates that is condiderably more fine-grain that previous estimations. Our experiments show that the loss rate predicted by our model is much more accurate than previous estimations.

1 Introduction

Large-scale distributed storage systems such as DHTs [1,3,4,2] can provide a low-cost alternative to expensive persistent storage solutions such as Storage Area Networks (SANs) and dedicated servers. DHTs guarantee persistence by replicating the same data object on several nodes, and regenerating replicas that are lost after a disk crash. However, unlike local-area storage systems, object replicas in a DHT are usually stored on geographically dispersed nodes, so communications between replicas have usually low bandwidth.

A low-bandwidth environment limits the rate at which lost replicas can be regenerated. For instance, if each node stores 100 GB, has a 1 Mbit/s connection to the Internet, and allocates one third of its total bandwidth to restore failed replicas, then repairing a crashed hard disk will take approximately one month. Long repair times, in turn, increase the probability of permanent data loss. Assuming random failures, there's a non-zero probability that all replicas of a given object will fail within a short period of time. Since it takes so long to restore the whole contents of a failed disk, the last replica of an object may be lost before the system can restore at least one copy of it. Adding more replicas decreases the probability of irrecoverable data loss, but can never completely eliminate it.

In practice, the number of replicas per object is chosen so that the probability of data survival is kept above a desired value. However, care must be taken when choosing the replication factor. An excessively high value will reduce the usable

T. Janowski and H. Mohanty (Eds.): ICDCIT 2007, LNCS 4882, pp. 184–196, 2007.

storage capacity and increase the network overhead. Conversely, a low replication factor may result in poor durability.

Recent work has shown that object durability can be predicted by modeling the state of the system using a continuous-time Markov chain [6,7]. The chain models the number of replicas of a given object that are present in the system at a given time. Although this model is relatively simple, the difficulty resides in estimating the chain's transition rates accurately. Incorrect estimations of these model parameters will result in the probability of data survival being underestimated, or, worse, overestimated. This, in turn, will lead the system's designer into choosing a replication factor based on poor model predictions.

Although the failure rate may be estimated rather easily (using the disk MTBF, for instance), modeling the repair rate is much more difficult, as this depends on a myriad of factors such as the number of available replicas, the amount of data per node, the available bandwidth, and even the failure rate. In this paper we provide an analytical expression for system's repair rate that takes into account all these factors.

This paper makes the following contributions. First, it shows that estimating the Markov chain repair rates to predict durability in DHTs is a hard problem, as these rates depend on a large number of factors. Second, it presents an analytically-derived expression of the system's repair rate that is much more accurate than previous estimations. This increased accuracy directly translates into a much better prediction of the probability of object loss in the system.

The rest of this paper is organized as follows. Section 2 lists our assumptions and describes the Markov chain model. Section 3 discusses previous approximations of the chain repair rates, and presents our new analytical expression to estimate these rates. Section 4 evaluates the model's predictions through long-term simulations, and Section 5 concludes the paper.

2 Model

2.1 Assumptions and Definitions

We consider a Distributed Hash Table composed of thousands of nodes connected to the Internet by low-bandwidth links, e.g., 1 Mbit/s links. Each object is associated with a unique key, and is replicated on k adjacent nodes which are close to the key in the DHT address space. This is a common replication scheme used by DHTs such as PAST [1] and OpenDHT [14]. We assume a ring address space as this is the easiest to visualize, but our results also apply to other geometries such as XOR [3] and d-torus [5]. We assume that each node stores thousands of objects, and a capacity from tenths to hundreds of gigabytes per node.

The contents of a node may be lost due to a hard disk crash or a destructive operating system reinstall. We make no difference between the two, and we will use the terms node *failure* and *crash* to refer to the same event. We also assume that failures are random and uncorrelated. We assume that failed hard disks are replaced by an empty disk quickly after the crash (i.e., within a few

hours). We ignore this replacement time as it is negligible compared to the time needed to regenerate a disk's contents. The node then starts regenerating the objects it stored before the crash using the following repair procedure: first, since replicas are stored on ring neighbors, the node determines which objects are to be restored by querying its neighbors. Then, the node starts transferring the objects sequentially from the remaining replicas. This is basically what existing DHTs do to regenerate replicas after a crash.

We assume highly stable nodes, i.e., that the disconnection and churn rate are low. Contrary to P2P file-sharing systems, which are characterized by high churn [8,9], storage systems must rely on nodes which stay connected for long periods of time (i.e., weeks or months) in order to guarantee data persistence [10]. These could be, for example, the workstations of a corporate network (in a system that federates several networks), users who leave their P2P client running continously, or residential set-top boxes equipped with hard disks and running a P2P storage service. Because of this, churn and temporary disconnections will not be considered in our current analysis. A more refined model that takes these into account is left for future work. Also, since we assume high availability, our analysis will only consider systems that use replication, rather than erasure codes, as the former has been shown to be more efficient for highly available nodes [16].

Finally, we list some important definitions that will be used throughout the following sections:

- MTBF. Mean time between failures of a given node. This figure may be smaller than the hardware failure rate of a disk because of destructive operating system reinstalls.
- b. Average number of bytes stored per node.
- bw_{max}. Maximum bandwidth per node allocated to regenerate lost replicas. We assume symmetric upstream and downstream bandwidths.
- θ. The value of θ gives the ratio between the MTBF and the minimum time needed to restore the contents of a hard disk. It is defined as follows:

$$\theta = \frac{MTBF}{b/bw_{max}}$$

  θ is a key parameter of our model. High values of θ indicate that a node will finish restoring its hard disk long before it crashes again, and that it only spends a small fraction of its total uptime regenerating replicas lost after a crash. Conversely, a small θ means that is likely that the node will fail again shortly after it finishes restoring its disk from the previous crash, or even before the disk is completely restored.

2.2 Markov Chains

In order to estimate data durability, we model the state of a data object using a continuous-time Markov chain [6,7]. Each state in the chain represents the number of replicas for a particular object that exist in the system at time t after

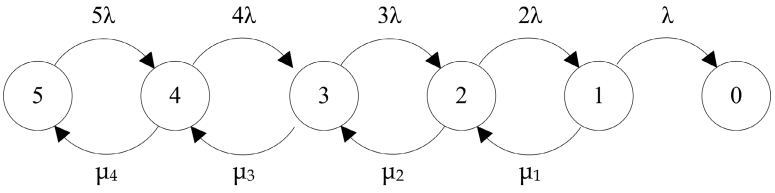

Fig. 1. The number of live replicas of a given object is modeled using a continuous-time Markov chain

its insertion into the DHT. Therefore, the chain has $k+1$ states, 0 to k, where k is the replication factor. Figure 1 shows the Markov chain for $k = 5$. State transitions correspond to replicas being lost or restored. Once state 0 is reached, there are no more copies left and the object is lost. The probability of being in state i as a function of the time t can be obtained by solving a system of $k+1$ linear differential equations. The probability of losing the object permanently is simply the probability of reaching state 0 at time t.

Failure rates, i.e., transitions to lower states, are caused by hard disk crashes or system reinstalls, so they can be approximated using the disk MTBF provided by the manufacturer or the destructive reinstall rate observed on the real system. As usual, we will assume that the time between failures of the same node is exponentially distributed with mean MTBF$= \lambda^{-1}$ [11,6,7]. Thus, if each disk fails with MTBF, then the time between two failures in a set of i independent disks becomes MTBF$/i = (i\lambda)^{-1}$. This yields a failure rate $i\lambda$ from state i to state $i-1$.

The repair rate μ_i, i.e., from state i to state $i+1$, is much harder to determine. Intuitively, it corresponds to the inverse of the mean time needed by the system to restore a *single* copy of the object when $k-i$ replicas are missing. However, the average time needed to regenerate *one* replica is not independent of the number of replicas being repaired. As the number of failed replicas increases, fewer sources become available to download from. This means that two or more nodes may contact the same source, congesting its upstream link and decreasing the average transfer rate.

3 Analytical Expression for the Transitions Rates μ_i

3.1 Previous Estimations

Chun et al. [7] have presented a system for persistent data storage called Carbonite, in which they predict durability using the same model of Figure 1. They suggest using a constant repair rate $\mu_i = \mu'$, and propose a simple approximation for μ'. Their estimation for the value of μ' is as follows[1]: if each node stores

[1] We refer to this value as μ' to differentiate it from our estimation of μ which will be presented in the next section.

b bytes of data in its hard disk, and has allocated a maximum repair bandwidth of bw_{max} bytes/sec, then the time needed to restore a crashed hard disk is $T'_r = b/bw_{max}$. Therefore, the model repair rates are:

$$\mu'_i = \mu' = \frac{1}{T'_r} = \frac{bw_{max}}{b} \tag{1}$$

In a different approach, Ramabhadran et al. [6] suggest that the repair rate μ_i grows with the number of missing replicas, and propose the following linear expression:

$$\mu''_i = (k - i)\mu \tag{2}$$

However, the authors do not provide any expression for μ, as they consider it to be a tunable system parameter, dependent on how aggressively the system reacts to failures.

As we will see in the next sections, the expression for μ' is too simple and provides a poor estimation of the real mean repair rate. The expression for μ'' assumes a parametrable repair rate, but the authors do not provide any expression for its maximum value.

3.2 Theoretical Analysis

In this section we present a new analytical expression for the mean repair rate μ. The value of μ represents the rate at which, in average, one copy of a given object is regenerated after a crash. We then use this value to approximate μ_i using the linear equation 2 presented in Section 3.1.

Estimating μ. We define μ as the mean rate at which one object replica is restored after a disk crash, or equivalently, as the inverse of the mean time $\overline{t_r}$ required to restore an object replica:

$$\mu = \frac{1}{\overline{t_r}} \tag{3}$$

To estimate $\overline{t_r}$, we start by noticing that each hard disk does not store a single object of b bytes, but rather thousands of smaller objects. Thus, assuming that the crash occurs at t_{crash}, that restoring the contents of the hard disk takes a time T_r, and that the node does not crash again while the disk is being restored, then each object i will be restored at some time $t_{crash} + t_{r,i}$, with $0 < t_{r,i} \leq T_r$. Second, when more than one replica of the same object is missing, it becomes highly likely that the remaining replicas will receive concurrent download requests from several nodes. Since the uploader's upstream bandwidth must be shared by several downloaders, the effective bandwidth available to each restoring node will be lower than the maximum bandwidth bw_{max}.

Finally, depending on the amount of data stored per node, the time needed to restore a hard disk may not be negligible compared to the MTBF. In such cases (i.e., for small values of θ), there is a small but non-zero probability that the same

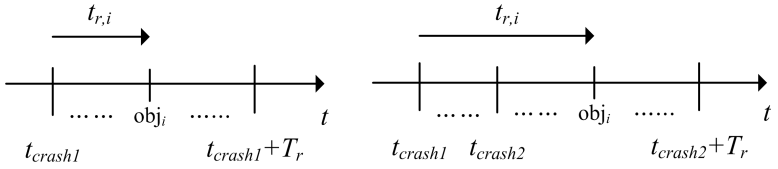

Fig. 2. A premature crash occurs when the node fails before it has finished restoring its hard disk. As a result, the repair time for that object is significantly increased.

node will fail a second time before it has finished restoring the contents of its hard disk, as shown in Figure 2. The probability of premature crashes depends on the value of θ. For $\theta \gg 1$, the hard disk restore time is negligible compared to the mean time between failures, and premature crashes will be rare. In practice, however, the system may exhibit values of θ for which premature crashes cannot be ignored. For instance, Chun et at. show that storing 500 GB per node on Planetlab nodes yields $\theta = 6.85$ [7]. With this value of θ, the probability of a premature crash is around 14%.

Analytical expression for $\overline{t_r}$. We now obtain an analytical expression for $\overline{t_r}$ by taking into account the issues described above. Our theoretical analysis is based on the following key assumption: the nodes that are restoring their hard disk consume a portion of the bandwidth allocated for replica regeneration. Therefore, when a node contacts another node to transfer an object back to its hard disk, the available bandwidth will be less than bw_{max}.

Before we detail our analysis, a few definitions are in order:

- T_r. Time required to restore the whole contents of a hard disk (i.e., b bytes), assuming that no premature crashes occur during the restore process. This is not simply b/bw_{max}, as the actual available bandwidth will be lower than bw_{max} due to nodes downloading from the same sources concurrently.
- b_e. Average amount of bytes transferred per node between two successive crashes. For $\theta \gg 1$, we have $b_e \approx b$. However, for smaller values of θ we have $b_e < b$ due to the effect of premature crashes. In fact, when a premature crash occurs, the amount of bytes transferred between those two crashes will be less than b (e.g., between t_{crash1} and t_{crash2} in Figure 2), resulting in an average value which is lower than b.
- bw_b. Average background bandwidth consumed by each node due to replica regeneration. This is simply the average amount of bytes b_e transferred between crashes, divided by the MTBF:

$$bw_b = \frac{b_e}{MTBF} \tag{4}$$

- bw_r. Effective bandwidth available to each node for replica regeneration. This is obtained by substracting the background bandwidth bw_b consumed by other nodes from the maximum repair bandwidth bw_{max}:

$$bw_r = bw_{max} - bw_b \tag{5}$$

– X. Random variable representing the time between two successive crashes of the same node. X follows an exponential distribution with mean MTBF=λ^{-1}:

$$f_X(x) = \lambda e^{-\lambda x} \text{ for } x \geq 0$$

The following analysis contains two parts. First, we find the value of T_r as a function of the system parameters b, bw_{max}, MTBF. Second, we obtain an expression for $\overline{t_r}$ as a function of T_r.

Let $Z = \tau(X)$ be the time that the node spends downloading objects between two successive crashes separated by a time X. We then have:

$$Z = \tau(X) = \begin{cases} X \text{ for } X < T_r \\ T_r \text{ for } X > T_r \end{cases}$$

Clearly, if $X < T_r$ then a premature crash has occurred, so the time spent downloading objects is equal to the time between the two crashes. Conversely, for $X > T_r$ the hard disk is completely restored in a time T_r, after which the node remains idle until the next crash.

Since X is exponentially distributed with parameter λ, the expected value of Z, which we will call T_e, is:

$$T_e = \int_0^\infty \tau(x) f_X(x) dx = \int_0^{T_r} x\lambda e^{-\lambda x} dx + \int_{T_r}^\infty T_r \lambda e^{-\lambda x} dx = \frac{1 - e^{-\lambda T_r}}{\lambda} \quad (6)$$

The value of T_e can be interpreted as the average time a node spends transferring objects between two consecutive crashes, taking into account the probability of premature crashes. Notice from equation 6 that T_e depends on T_r, which is not known yet.

We then notice that b_e is the amount of bytes transferred during the time interval T_e using a repair bandwidth bw_r. Similarly, a node will transfer b bytes during a time T_r using a bandwidth bw_r. Therefore, we have:

$$bw_r = \frac{b_e}{T_e} = \frac{b}{T_r} \Longrightarrow b_e = \frac{T_e}{T_r} b \quad (7)$$

Combining equations 4, 5 and 7 we have the following two equations:

$$bw_b = \frac{b_e}{MTBF} = \frac{T_e}{T_r} \frac{b}{MTBF} \quad (8)$$

$$T_r = \frac{b}{bw_r} = \frac{b}{bw_{max} - bw_b} \quad (9)$$

Notice that in equations 6, 8 and 9 the only unknown variables are T_e, bw_b and T_r (b, bw_{max}, and MTBF are known). Therefore, we can find their values by solving a system of three equations. This can only be done numerically, as T_r appears in the exponent of equation 6.

From this point we will assume that the value of T_r is known (as it can be computed numerically from the system of equations we just described). We will now use T_r to find an expression for $\overline{t_r}$.

Let Y be a random variable representing the time at which a given object is restored after a given crash. According to this definition, the mean object repair time $\overline{t_r}$ is simply the expected value of Y, i.e., $\overline{t_r} = E[Y]$.

In order to find $E[Y]$, we must consider several cases:

1. The node finishes restoring its disk before the next crash occurs, i.e., $X > T_r$. In this case, the object will be restored at some time y with $0 < y < T_r$. Assuming that the variable Y is uniformly distributed in the interval $[0, T_r]$, the average object restore time for this case is:

$$\overline{t_{r1}}(x) = E[Y|X = x, x > T_r] = \frac{T_r}{2}$$

2. A premature crash occurs, i.e., $X < T_r$. We must distinguish between two more cases:
   (a) The object is restored before the node crashes again, i.e., $Y < X$. The probability of this occuring, given that a premature crash has taken place at $t = x$, is:

$$p_{2a}(x) = P(Y < X|X = x, x < T_r \wedge x > Y) = \frac{x}{T_r}$$

   In this case, the average object repair time will be uniformly distributed in the interval $[0, x]$. Thus, we have:

$$\overline{t_{r2a}}(x) = E[Y|X = x, x < T_r \wedge x > Y] = \frac{x}{2}$$

   (b) The node crashes again before the object is restored, i.e., $Y > X$. As before, we obtain the probability that this occurs:

$$\begin{aligned} p_{2b}(x) &= P(Y > X|X = x, x < T_r \wedge x < Y) \\ &= 1 - P(Y < X|X = x, x < T_r \wedge x > Y) = 1 - \frac{x}{T_r} \end{aligned}$$

   In this case, however, the node crashes again before the object is repaired, and restarts a new repair procedure from stratch. All we can say is that the object will be repaired after some average time $\overline{t_r} = E[Y]$ from the beginning of this new repair procedure. Thus, in this case we have:

$$\overline{t_{r2b}}(x) = E[Y|X = x, x < T_r \wedge x < Y] = x + \overline{t_r}$$

Finally, the mean object repair time $\overline{t_r}$ is obtained as the expected value of Y:

$$\begin{aligned} \overline{t_r} = E[Y] &= \int_0^\infty E[Y|X = x] f_X(x) dx \\ &= \int_0^{T_r} \left[\overline{t_{r2a}}(x) p_{2a}(x) + \overline{t_{r2b}}(x) p_{2b}(x) \right] f_X(x) dx + \int_{T_r}^\infty \overline{t_{r1}}(x) f_X(x) dx \\ &= \int_0^{T_r} \left[\frac{x}{2} \frac{x}{T_r} + (x + \overline{t_r})(1 - \frac{x}{T_r}) \right] \lambda e^{-\lambda x} dx + \int_{T_r}^\infty \frac{T_r}{2} \lambda e^{-\lambda x} dx \end{aligned}$$

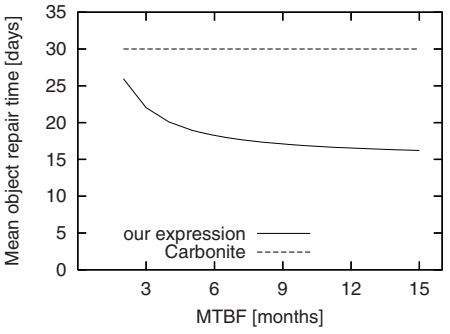

Fig. 3. Mean repair time predicted by our expression, as well as Carbonite's, for different values of the MTBF

After computing the integrals, we obtain $\overline{t_r}$ as a function of λ and T_r:

$$\overline{t_r} = \frac{1}{\mu} = \frac{1 + e^{\lambda T_r}(\lambda T_r - 1)}{\lambda(e^{\lambda T_r} - 1)} \tag{10}$$

Figure 3 shows the variation of the mean repair time (i.e., $\overline{t_r} = 1/\mu$) as a function of the MTBF for a system storing 300 GB per node and allocating 1 MBit/s per node as maximum repair bandwidth bw_{max}. Notice that Carbonite's expression for μ does not depend on the MTBF (cf. equation 1), thus resulting in the constant value shown in Figure 3.

First-order approximation of μ_i. Although we have obtained an analytical expression for the value of μ, the Markov chain requires $k - 1$ repair rates, one for each state i in the chain, with $0 < i < k$. As a first approximation, we will consider that μ_i increases linearly with each state, i.e., $\mu_i = (k - i)\mu$. Notice that this linear form has already been suggested by other authors (cf. equation 2). However, their model lacked a general expression for μ, considering it as a tunable parameter. As we will see in the next section, simulations show that a linear expression provides a good approximation of the real μ_i when the system uses a small number of replicas (i.e., $k = 3$).

4 Validation

In order to evaluate the expressions of Section 3, we use a discrete-event simulator that implements a simplified version of the PAST protocol [1]. Each node has a unique identifier in the integer interval $[0, 2^{160}]$, thus adopting a ring geometry. Data objects are replicated on the k nodes which are closest to the object key.

We assume a symmetric repair bandwidth of 1.5 Mbit/s, as was previously done by other authors [7]. Each node stores b bytes, which can vary from tenths to hundreds of gigabytes of data, according to the experiment. In all cases the

data stored by each node is divided into 1000 objects, so each object has a size $b/1000$ bytes. We assume high node availability (cf. Section 2.1), so temporary node disconnections are ignored by our simulator.

All our simulations use a DHT of 100 nodes, as our simulations show that increasing the network size does not qualitatively change the results. The reason is that objects are replicated on adjacent nodes on the ring, so restoring a node's hard disk only affects a small number of other nodes. We generate synthetic traces of failures by obtaining node inter-failure times from an exponential distribution of mean MTBF [7]. Whenever the last replica of an object is lost, the simulator logs the time elapsed since the insertion of that object into the DHT, and inserts a new object of the same size. This ensures that the amount data in the DHT remains constant during the experiment.

Unless otherwise noted, we use an MTBF of two months. Although this is small for a hardware failure rate, it is not far from the failure rate observed in a study conducted on Planetlab nodes [12]. Also, the durability of a system depends on the key parameter θ, i.e., the ratio between the MTBF and the node capacity. For instance, a system with $\theta = 10$ will exhibit the same durability whether it stores 100 GB per node with a MTBF of 2 months, 500 GB per node with an MTBF of 10 months, or any other equivalent combination of b and MTBF. By choosing a MTBF of 2 months and varying the node capacity b between 50 and 500 GB per node, we can test our model for $2 \leq \theta \leq 20$, thus covering the configurations most likely found in real systems.

4.1 Mean Repair Rate μ

In this experiment we measure the mean repair rate μ and compare it to that obtained through the analytical expression of Section 3.2. Each time a node crashes, we measure the time t_r it takes for each lost object replica to be recreated. Since transfers are serialized, some objects will be recreated shortly after the crash, whereas other will be pending until almost the end of the restore process of that node. We measure μ as the inverse of the average of all repair times t_r for all objects restored in the system. We use a constant MTBF of two months, and we vary the amount of data per node b from 50 GB ($\theta = 20$) to 500 GB ($\theta = 2$).

Figure 4 shows the mean object repair time $\overline{t_r} = 1/\mu$ measured by the simulator for $k = 3$ and $k = 7$, that obtained with our analytical expression, and the one produced by Carbonite's expression (cf. equation 1). The repair time increase with smaller values of θ, as this corresponds to a higher storage capacity per node. The figure also shows the error between the measured and predicted values. Our expression stays always within 20% of the measured μ, and in most cases within 5%. Conversely, Carbonite's value deviates considerably from the measured μ for large values of θ.

4.2 Probability of Object Loss

In Section 3.2 we suggested using the linear approximation $\mu_i = (k - i)\mu$, where μ is the value produced by our analytical expression. To evaluate the accuracy

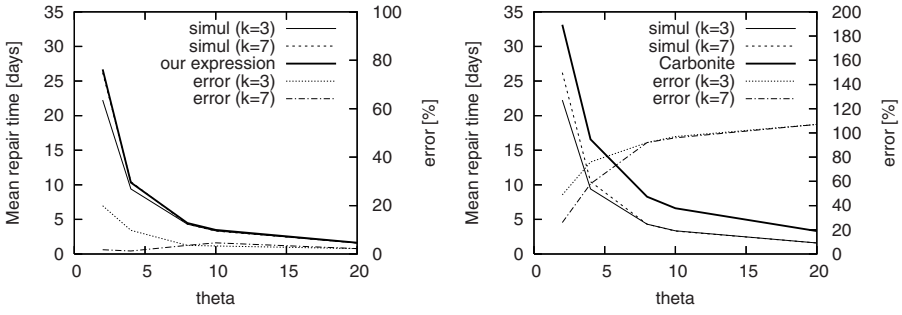

Fig. 4. Mean repair time measured by simulation, compared to the value predicted by our estimation (left) and Carbonite's expression (right)

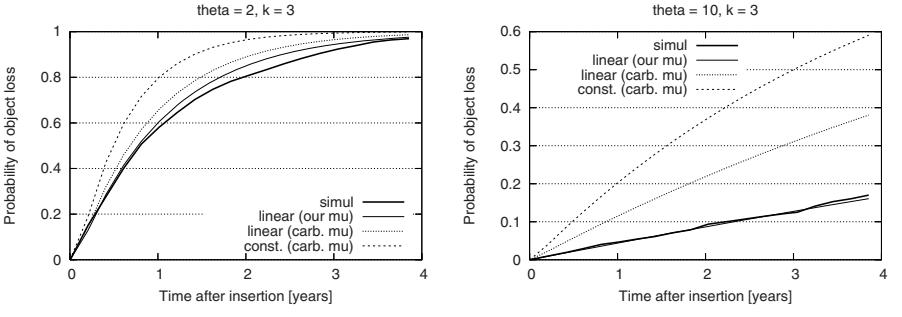

Fig. 5. Probability of object loss measured by simulation and predicted by the Markov chain

of this approximation, we measure the probability of object loss predicted by the Markov chain, and compare it with the loss rate observed on the simulator.

The loss rate predicted by the Markov chain is obtained by solving a system of $k+1$ differential equations. For simplicity, we find a numerical solution using Mathematica's NDSolve [13]. The loss probability measured by simulation is obtained by counting the percentage of objects that are lost at time t after their insertion. In order to smooth the curve we run 5 simulations and average the results.

Figure 5 shows the probability of object loss over time, using $k = 3$ and two values of θ. Our linear approximation $\mu_i = (k-i)\mu$ produces a good prediction of the probability of object loss over time. Conversely, using the value μ' suggested in the Carbonite paper [7] overestimates the probability of object loss, regardless of the approximation used to obtain μ_i (linear or constant). The error when using μ' is higher for the higher value of θ, which is consistent with Figure 4.

Unfortunately, for larger values of k (i.e., $k > 3$) the predicted loss rate is no longer accurate (the curves are not shown for space reasons). This is not due to our analytical estimation of μ, but to the first-order approximation of μ_i, which

produces poor results for $k > 3$. In fact, our measurements show that the values of μ_i grow sublinearly with i. We are currently working on deriving an analytical expression for the values of μ_i, which will prove useful for $k > 3$.

5 Conclusions and Future Work

In this paper we have focused on finding an accurate prediction of data durability in DHTs. Recent work has suggested that this can be achieved by modeling the system's behavior using Markov chains. However, our experiments show that the repair rates suggested previously produce inaccurate predictions of the object loss probability. We have presented a new analytical estimation for the mean repair rate which produces much more accurate results. We have also shown that a first-order approximation of the chain's repair rates yields good predictions for a small replication factor (i.e., $k = 3$). For higher values of k, a higher-order approximation must be found to produce accurate predictions of the system's durability. Future work will include considering the impact of churn, as well as deriving an analytical expression for all repair rates of the chain.

References

1. Rowstron, A., Druschel, P.: Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In: Proc. of SOSP (2001)
2. Rhea, S., Godfrey, B., Karp, B., Kubiatowicz, J., Ratnasamy, S., Shenker, S., Stoica, I., Yu, H.: OpenDHT: A Public DHT Service and Its Uses. In: Proc. of SIGCOMM (2005)
3. Maymounkov, P., Mazieres, D.: Kademlia: A Peer-to-peer Information Systems Based on the XOR Metric. In: Proceedings of IPTPS (2002)
4. Dabek, F., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I.: Wide-area cooperative storage with CFS. In: Proc. of SOSP (2001)
5. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content-Addressable Network. In: Proc. of SIGCOMM (2001)
6. Ramabhadran, S., Pasquale, J.: Analysis of long-running replicated systems. In: Proc. of INFOCOM (2006)
7. Chun, B., Dabek, F., Haeberlen, A., Sit, E., Weatherspoon, H., Kaashoek, F., Kubiatowicz, J., Morris, R.: Efficient Replica Maintenance for Distributed Storage Systems. In: Proc. of NSDI (2006)
8. Saroiu, S., Gummadi, P.K., Gribble, S.D.: A measurement study of peer-to-peer file sharing systems. In: Proc. of MMCN (2002)
9. Gummadi, K.P., Dunn, R.J., Saroiu, S., Gribble, S.D., Levy, H.M., Zahorjan, J.: Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In: Proc. of SOSP (2003)
10. Blake, C., Rodrigues, R.: High availability, scalable storage, dynamic peer networks: Pick two. In: Proc. of HotOS (2003)
11. Patterson, D., Gibson, G., Katz, R.: A case for redundant arrays of inexpensive disks (RAID). In: Proc. of SIGMOD (1988)
12. Dabek, F.: A Distributed Hash Table. Ph.D. thesis (2005)
13. Tam, P.: A Physicist's Guide to Mathematica. Elsevier, Amsterdam (1997)

14. Rhea, S.: OpenDHT: A Public DHT Service. Ph.D. thesis (2005)
15. Zhao, B., Huang, L., Stribling, J., Rhea, S., Joseph, A., Kubiatowicz, J.: Tapestry: A Resilient Global-Scale Overlay for Service Deployment. IEEE Journal on Selected Areas in Communications 22(1) (January 2004)
16. Rodrigues, R., Liskov, B.: High Availability in DHTs: Erasure Coding vs. Replication. In: Castro, M., van Renesse, R. (eds.) IPTPS 2005. LNCS, vol. 3640. Springer, Heidelberg (2005)