Storage and Network Resource Usage in Reactive and Proactive Replicated Storage Systems

Rossana Motta and Joseph Pasquale

Department of Computer Science and Engineering
University of California San Diego
La Jolla, CA, USA
Email: {rmotta, pasquale}@eng.ucsd.edu

***Abstract*—Replicated storage systems allow their stored data to outlive the life of the nodes storing them by use of replication. In such systems, failed replicas are "repaired" by copying remaining replicas from other nodes. We consider two main replication strategies: reactive, in which replication occurs in response to failures, and proactive, in which replication occurs in anticipation of failures. There is no consensus in the literature about which strategy is better. Our work presents a quantitative analysis that compares reactive and proactive through simulations and verified analytically where possible. In particular, we systematically compare how the strategy of replication affects the usage of storage and network resources, at peak consumption as well as on average, to achieve a given lifetime for a replicated object.**

Our results indicate that a proactive strategy leads to multiple times higher storage requirements than a reactive strategy, with the exact number depending on parameter values for failure and replication rates. Reactive systems are moderately bursty in terms of bandwidth consumption, with rare peaks of at most five times (for realistic parameter values) that of proactive systems, the latter of which is constant by design given that replications are periodic for proactive.

I. Introduction

Replicated storage systems are currently used in a wide variety of contexts to extend the lifetime of data objects beyond the lifetime of the node(s) hosting the data. For instance, cloud computing, peer-to-peer data-sharing systems, disaster recovering systems and paid storage services all heavily rely on distributed replication of the data they store.

In this work, we focus on *durability* in replicated storage systems, that is the duration for which an object is retrievable from the system. More precisely, we quantify durability by the *lifetime* of an object, which is the interval of time between the creation of the object and the time that it is permanently irretrievable from the system. The lifetime is extended by replicating, i.e., making copies, of the object, in whole (i.e., full replication) or in parts (e.g., via erasure coding). New replicas can only be created if there still are existing replicas on some node(s) in the system; if there are not, then the object's lifetime is ended at the point in time when there are no more replicas.

Replicated storage systems use two main strategies of replication:

- *Reactive*: Each failure triggers a repair, that is the creation of a new replica;
- *Proactive*: Replications occur in anticipation of failures, typically in a periodic fashion.

A major open question regarding these strategies is: Which one is better, in terms of producing longer average lifetimes, minimizing resources used, etc.? Our objective is to quantify the differences between reactive and proactive models through their respective resource consumption, particularly the storage used for replicas, and the bandwidth used to transfer copies of replicas. Our models are kept simple to isolate the core replication strategies (and thus we do not include various advanced features found in real systems), as that is what we want to especially focus on in our analysis.

In the remainder of this paper, we describe Related Work in Section II, Models and Assumptions in Section III, Performance Results of Reactive vs. Proactive in Section IV, and Conclusions in Section V.

II. Related Work

A number of distributed storage systems have been proposed. Among those implemented on real nodes or simulated, they can be classified as reactive, such as Oceanstore [9], Glacier [7], Carbonite [6], TotalRecall [2] and Thriftstore [5], or proactive, such as Phoenix [8] and Tempo [1]. Oceanstore, TotalRecall and Phoenix focus on availability. The others focus on durability.

Each system has unique features. Oceanstore opportunistically caches objects in a nomadic fashion, overcoming lack of trust through cryptography. Glacier focuses on erasure coding, dealing with massive correlated failures and traffic due to replication. Carbonite aims at minimizing the resource consumption by replicating only when the total number of replicas falls below a given threshold. TotalRecall automates the management of replication, allowing users to choose between full replication and erasure coding. and to specify a given availability target. Thriftstore allows one to choose either availability or durability as a system parameter. It utilizes the exponential distribution to model permanent failures and the Weibull distribution to model transient failures. Phoenix is a cooperative architecture for backup that focuses on the diversity of hosts to replicate in a way that correlated failures can be avoided and that objects can survive catastrophic events.

Tempo, similar to Carbonite, uses a threshold to stop replicating when the number of replicas reaches a given number. Tempo also presents a quantitative comparison with Carbonite, thus offering a rare comparison of a proactive system with a reactive one. A primary and important finding is that Tempo performs better than Carbonite in terms of bandwidth usage, as peaks are avoided.

Our primary differences with the above related works are that we analyze the core replication strategies in their most basic forms, and how they affect the expected lifetime, its variability, and resource requirements. We also assume that the system is fully distributed and thus there is no central authority that can monitor how many copies of an object there are in the system.

III. Models and Assumptions

We make the following simplifying assumptions in our models. These allow us to obtain results on interesting first-order behaviors while still making the models tractable and easy to understand.

- We consider a single replicated object whose lifetime is to be determined.
- A node can hold exactly one replica (thus, if the node fails, the replica "fails," i.e., it is permanently inaccessible).
- There are $n \geq 2$ initial nodes, where n depends on, or is a measure of, the available storage.
- We base our models on Markovian hypotheses, according to which the distributions of failure times for reactive and proactive are exponential with parameter λ (the mean failure rate). The distributions of repair times are also exponential for reactive, with parameter μ (the mean repair rate). However, in a proactive system, replications are periodic, occurring once every interval of time of length $1/\rho$, and so ρ is the replication rate.
- When we select values for λ and μ, we base them on measurement data from PlanetLab [3], a global research network consisting of more than 1000 nodes distributed over many hundreds of sites worldwide.

IV. Reactive vs. Proactive: Performance Results

A. Expected Lifetime When Reactive Repair Rate Equals Proactive Replication Rate

We first compare reactive and proactive in terms of the lifetimes they produce by making the mean reactive repair rate μ the same as the mean proactive replication rate ρ. Figure 1 shows a comparison of the achieved expected lifetime for reactive and proactive as a function of μ and ρ. It is clear that, especially as the value of the rate of repair or replication increases, a reactive system achieves much longer lifetimes compared to one that is proactive. In addition, as n increases, the difference between the achieved expected lifetimes in reactive and in proactive increases rapidly.

These results are due to the fact that, in reactive, repairs are performed when needed, whereas in proactive, replication is indifferent as to failures that happen in the system. Thus,

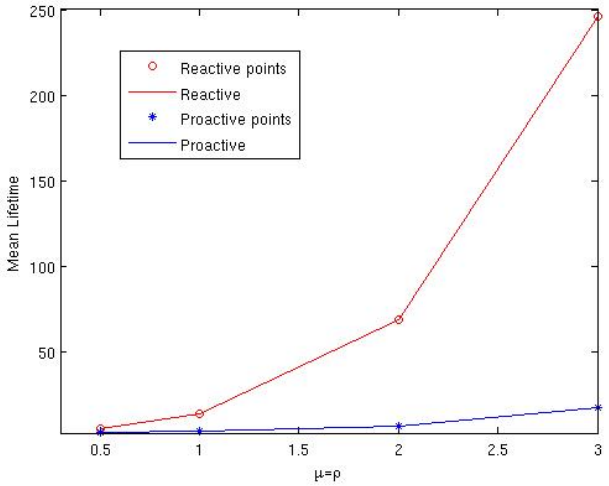


Fig. 1. Expected lifetime (computed by averaging over 1 million simulations) as a function of μ and ρ when μ=ρ. Here, n=6 and λ=1. When the reactive repair rate μ has the same value as the proactive replication rate ρ, the reactive model produces longer mean lifetimes. The gap in values increases as μ and ρ increase.

for any given values of n and λ, if one wants to achieve the same value of mean lifetime for a reactive and for a proactive system, the proactive replication rate ρ needs to be higher than the reactive repair rate μ. This is true for all values of n and λ that we tested (n ranged between 2 and 50, and λ ranged between $\mu/10$ and 10μ).

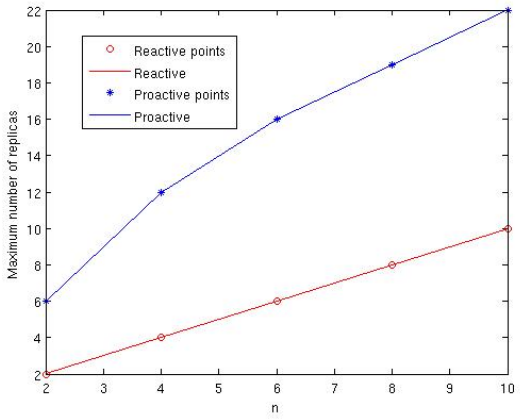
In the experiments that follow, whenever reactive and proactive are compared with each other, the comparison is carried out by using the same values of n and λ for proactive and reactive, while choosing the reactive repair rate μ and the proactive replication rate ρ so that the mean lifetime is the same (up to the second decimal digit) for reactive and proactive. As indicated above, to achieve the same mean lifetime, the value of ρ will generally be larger than the value of μ.

B. Storage Requirements

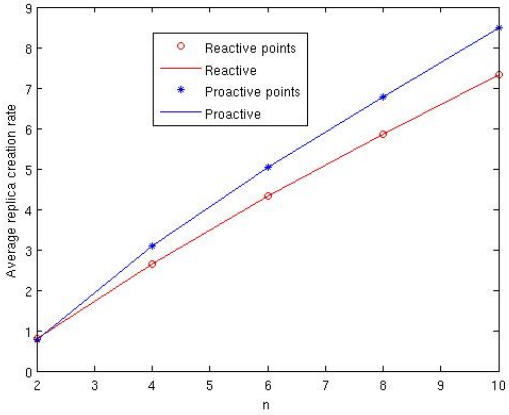
A basic question when comparing reactive and proactive models is: How much storage do they require, on average as well as at the maximum, to achieve the same level of expected lifetime? Such a comparison is significant in any real world scenario, as generally there are constraints imposed by the number of available nodes and by storage costs.

We compared reactive and proactive through the analysis of two parameters: the maximum number of replicas reached at some point during the lifetime of the object and the average replica creation rate, defined as the total number of created replicas divided by the lifetime of the object. The two parameters vary as n, λ, μ and ρ vary, thus making the problem multidimensional. Therefore, we restricted the study to likely real world scenarios, namely n varying between 2 and 10, and μ varying between 0.5λ and 4.5λ.

Figure 2 reports the maximum number of replicas and the

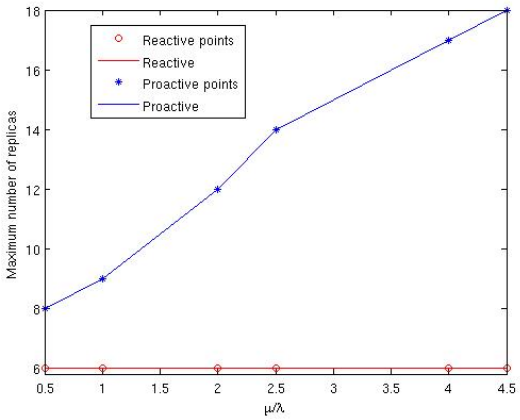


(a) Maximum number of replicas. For reactive, the maximum number of replicas is the same as n because a replication occurs only in response to a failure.

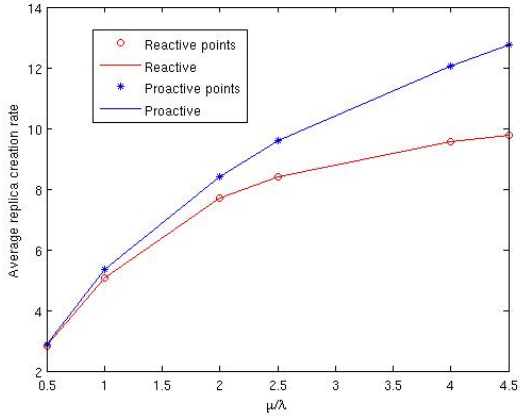


(b) Average replica creation rate, defined as the total number of created replicas divided by the lifetime of the object.

Fig. 2. Maximum number of replicas and average replica creation rate as functions of n. λ=1, μ=2.756, ρ=1.32-8.49. The plots show that proactive uses considerably more storage than reactive, especially when considering maximum storage usage.



(a) Maximum number of replicas. Since we keep n constant, reactive never reaches a maximum number of replicas that is larger than n. This is why the curve for reactive is a horizontal line.



(b) Average replica creation rate, defined as the total number of created replicas divided by the lifetime of the object.

Fig. 3. Maximum number of replicas and average replica creation rate as functions of μ/λ. λ=2, n=6, ρ=3.3-12.76 The points are averages from 100,000 rounds of simulations, which makes them statistically significant. The plots again show that proactive uses considerably more storage than reactive, especially in terms of maximum storage usage.

average replica creation rate averaged over 100,000 simulations, in the reactive and in the proactive models, as a function of n, with λ=1 and μ=2.756. Given n, λ and μ, the proactive replication rate ρ is determined so that the mean lifetime in proactive is the same (up to the second decimal digit) as the mean lifetime in reactive. The replica creation rate was computed measuring the total created replicas in one simulation and dividing this quantity by the mean lifetime for that simulation. To achieve statistical significance, the simulations were repeated 100,000 times and each point in Figure 2 represents the average of 100,000 rounds of simulations.

Figure 3 again shows the maximum number of replicas and the average replica creation rate, but this time as a function of μ/λ, for n=6. In a real system, the failure rate λ is typically an inherent characteristic and thus a given. For this reason, we assume λ as constant and equal to 2, and change the value of μ only. The choice of the value *2* for λ is based on data from PlanetLab [10], where the average failure rate is a little over one failure/week. We round up to 2 for a more aggressive scenario.

Plotting the maximum number of replicas and the average replica creation rate both as a function of n (Figure 2) and as a function of γ (Figure 3) shows that proactive uses considerably more storage than reactive, especially when considering the maximum, when the usage in proactive is multiple times that in reactive. How much more storage is used depends on the specific values of the parameters, as the storage is affected by each of them, namely n, λ, μ and ρ.

To further analyze the maximum storage requirements in proactive, in Figure 4 we show the ratio between the maximum number of concurrently active replicas and the initial number of replicas, as a function of ρ, for n=2 and λ=2. As the repair rate μ grows in reactive, the replication rate ρ in proactive

must also grow, to achieve the same value of mean lifetime. As the replication rate grows, the number of concurrently active replicas also tends to grow. We show a plot with a very low value of n because this is the case that maximizes the ratio between the maximum number of concurrently active replicas and the initial number of replicas in proactive.

Our simulations show that the proactive model produces a higher number of concurrently active replicas, which is almost always greater than the initial value of *n*, to which the reactive model is inherently capped. The ratio between the maximum number of replicas and the initial number of replicas progressively increases in proactive as the failure rate becomes smaller than the repair and replication rates. This is because, in this case, replications keep occurring but failures happen more slowly, and replicas accumulate.

One could implement a system-wide check to cap the total number of replicas to a given number in a proactive system [1]. This comes at the price of either having at least one central authority that maintains knowledge of the entire system at any point in time and is able to communicate with each node. Or, one could also implement a fully decentralized strategy to monitor the existing number of replicas in a system, but a frequent, system-wide communication may introduce significant network overhead.

The replica creation rate varies considerably according to the system parameters. For values of the failure rate λ being multiple times smaller than the repair rate μ, which is the most realistic scenario, proactive produces a higher average replica creation rate than reactive. On the other hand, for λ larger than μ, the average replica creation rate tends to be lower in proactive than in reactive. This, however, is an unrealistic scenario, as the mean time to repair a component is typically shorter than its mean time to failure.

Overall, our studies show that in likely scenarios, opting for a proactive system will cost multiple times more storage compared to the same system implemented with a reactive replication strategy.

C. Bandwidth Requirements

Along with storage requirements, bandwidth requirements represent a fundamental aspect of comparison between reactive and proactive models. It has been pointed out that one of the main benefits of using a proactive system is a smoother (i.e., more regular) usage of bandwidth, as the replications are more evenly distributed over time [1], [11], [4]. In the following, we present an analysis of the burstiness of reactive models, to quantify the difference with bandwidth usage in proactive models.

Figure 5a shows a comparison between replica creation over time in reactive (left) vs. proactive (right). Figure 5b shows the corresponding histograms of replica creation every $1/\rho$. Each point in Figure 5a represents a reactive repair or a proactive replication and the two plots zoom in on regions of points that are representative of their respective entire curves. In the various combinations of parameters that we analyzed and that are typical of real systems ($\mu < \lambda$), the maximum number of

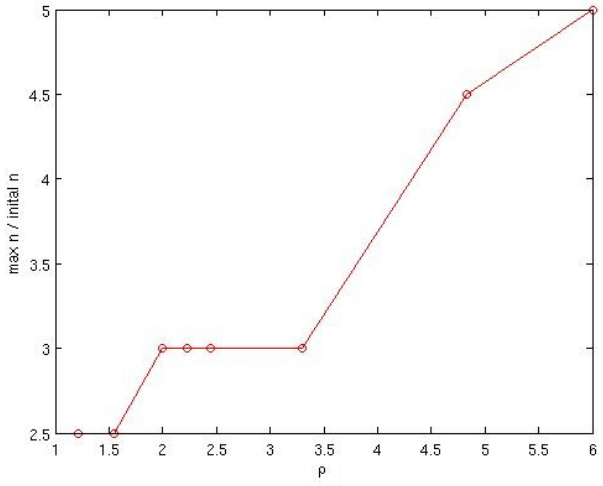


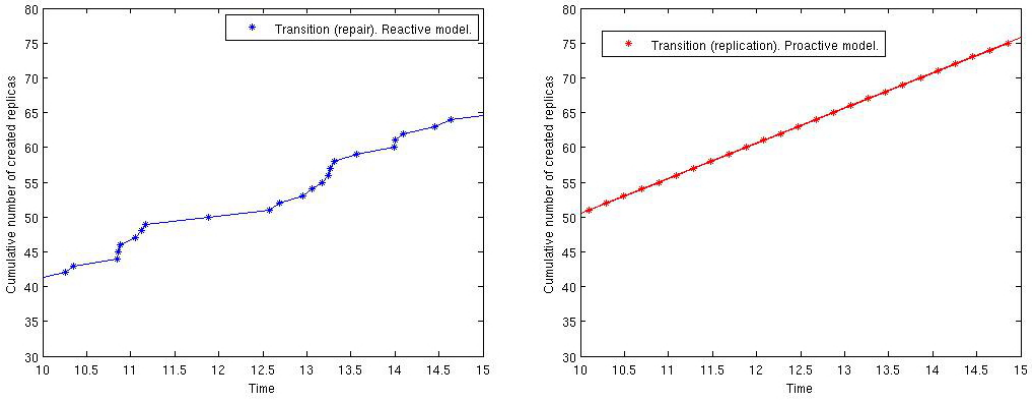
Fig. 4. Ratio between the maximum number of concurrently active replicas and initial number of replicas as a function of the replication rate ρ , in proactive, for n=2 and λ=2. As ρ increases, the ratio also increases.

replicas per $1/\rho$ is about five and the most recurrent case is one replica created each interval, as the histogram has a wide spike around the value 1. The burstiness increases as *n*, λ and μ increase, because as the three parameters increase, there are more nodes that fail and get repaired per unit of time. In Figure 5a, a highly bursty pattern in the replica creation for the reactive model would correspond to many points connected by lines with very high slopes. Our data suggest that the distribution of replica creation over time shows a moderately bursty pattern. The burstiness occurs rarely, and even when it occurs, the reactive bandwidth usage has peaks of roughly five times the usage in reactive.

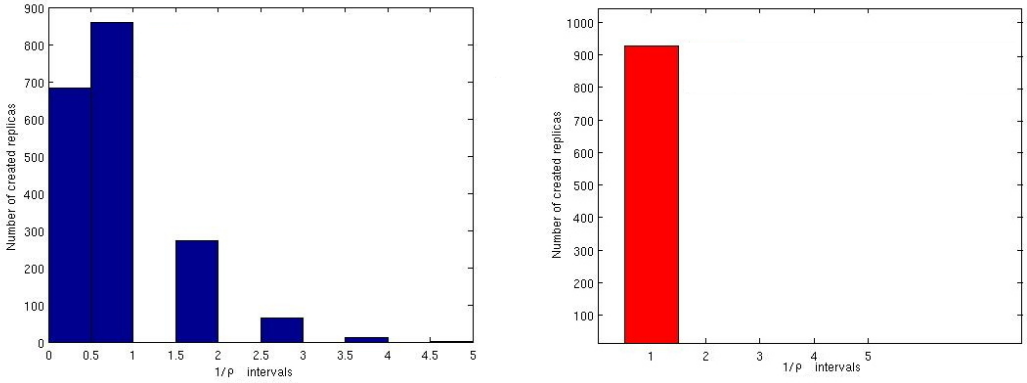
These results are most likely due to the fact that failures occur randomly and, since we are only considering uncorrelated failures, their distribution in time is random and relatively even. Since it is unlikely that bursts of uncorrelated failures occur within a short amount of time, bursts of repairs are also not observed. Additionally, when the failure rate λ increases, both μ and ρ must also be increased for the expected lifetime not to decrease. This implies that the time intervals $1/\rho$ become shorter. In other words, in proactive the replication must be more frequent and in reactive the probability of having many failures and repairs within each $1/\rho$ interval decreases, each interval being shorter. Thus, even when the failure rate increases, the reactive model does not appear to have a dramatically different bandwidth usage pattern compared with proactive.

V. Conclusions

The optimal strategy for replication between reactive and proactive to ensure durability in replicated storage systems is still an open problem. In this work, we considered a basic reactive system with exponential failures and exponential repairs, and a proactive system with exponential failures and constant periodic replications. We presented a quantitative



(a) Replica creation over time in reactive (left) and proactive (right). Reactive shows a moderately bursty pattern. For proactive, replications occur at a constant rate by design.



(b) Corresponding histograms of replica creation every $1/\rho$ intervals

Fig. 5. Replica creation in reactive (left) and proactive (right). For proactive, by design exactly one replica is created every $1/\rho$ interval, while for reactive, frequency of replications depends on frequency of failures. The most common case, however, is about one repair per $1/\rho$ interval. n=6, λ=1, μ=2.756 and ρ=5.05.

comparison of reactive and proactive strategies, specifically in terms of maximum and average resource usage. This gives insight in choosing the best strategy when designing a system, considering constraints of available bandwidth and storage. For realistic scenarios, our tests showed that whenever a system cannot tolerate any spikes in bandwidth greater than about five times the average available bandwidth, then a proactive strategy is likely the more appropriate. On the other hand, whenever storage constraints prevent the commitment of multiple times the number of initial replicas, a reactive model is the better choice.

REFERENCES

[1] E. S. Andreas, A. Haeberlen, F. Dabek, B. G. Chun, H. Weatherspoon, R. Morris, M. F. Kaashoek, and J. Kubiatowicz. Proactive replication for data durability. In *Proc. of the 5th Intl. Workshop on Peer-to-Peer Systems IPTPS*, 2006.

[2] R. Bhagwan, K. Tati, Y. chung Cheng, S. Savage, and G. M. Voelker. Totalrecall: System support for automated availability management. In *Proc. of the 1st ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 337–350, 2004.

[3] A. Chandra, S. Jain, and Z. Zhang. Scale systems: A systematic study of failures in planetlab. *Tech. Report*, 2008.

[4] A. Duminuco, E. Biersack, and T. En-Najjary. Proactive replication in distributed storage systems using machine availability estimation. In *Proc. of the 2007 ACM CoNEXT Conference*, pages 27:1–27:12, 2007.

[5] A. Gharaibeh and S. Al-Kiswany. Thriftstore: Finessing reliability tradeoffs in replicated storage systems. In *IEEE Transactions on Parallel and Distributed Systems, Vol. 22 No. 6*, 2011.

[6] B. gon Chun, F. Dabek, A. Haeberlen, E. Sit, H. Weatherspoon, M. F. Kaashoek, J. Kubiatowicz, and R. Morris. Efficient replica maintenance for distributed storage systems. In *Proc. of the 3rd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 45–58, 2006.

[7] A. Haeberlen, A. Mislove, and P. Druschel. Glacier: Highly durable, decentralized storage despite massive correlated failures. In *Proc. of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2005.

[8] F. Junqueira, R. Bhagwan, R. Hevia, K. Marzullo, and G. M. Voelker. Surviving internet catastrophes. In *Proc. of the Usenix Annual Technical Conference*, 2005.

[9] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *ACM SIGARCH Computer Architecture News*, volume 28, pages 190–201, 2000.

[10] S. Ramabhadran and J. Pasquale. Analysis of long-running replicated systems. In *Proc. of the 25th IEEE Annual Conference on Computer Communications INFOCOM*, 2006.

[11] H. Weatherspoon, H. Miranda, K. Iwanicki, A. Ghodsi, and Y. Busnel. Gossip: Gossiping over storage systems is practical. In *ACM SIGOPS Operating Systems Review*, volume 41, 2007.