Simulation Modeling of Weak-consistency Protocols

Richard A. Golding[†] *and Darrell D. E. Long*[‡]
Computer and Information Sciences Board
University of California, Santa Cruz
Santa Cruz, CA 95064

Abstract: Weak-consistency replication protocols can be used to build wide-area services that are scalable, fault-tolerant, and useful for mobile computer systems. We have evaluated the *timestamped anti-entropy* protocol to determine how well it meets these goals. In this protocol, pairs of replicas periodically exchange update messages; in this way updates eventually propagate to all replicas. In this paper we present results from a detailed simulation analysis of the fault tolerance and the consistency provided by this protocol. The protocol is extremely robust in the face of site and network failure, and it scales well to large numbers of replicas.

Keywords: wide-area communication, scalability, replication, fault tolerance.

1 Introduction

We are investigating an architecture for building distributed services that emphasizes scalability and fault tolerance. This allows applications to respond gracefully to changes in demand and to site and network failure. It also provides a single mechanism to support wide-area services and mobile computing systems.

The architecture uses data *replication* to meet availability demands and enable scalability. The replication is *dynamic* in that new servers can be added or removed to accommodate changes in demand. The system is *asynchronous*, and servers are as *independent* as possible; it never requires synchronous cooperation of large numbers of sites. This improves its ability to handle both communication and site failures. It uses large numbers of replicas, so replicas can be placed near clients and to spread query load over more sites.

Existing strong-consistency replication protocols only work well with small numbers of replicas. As their population increases, availability and performance decrease beyond the level acceptable for interactive applications, because they require the synchronous cooperation of a large number of sites. Instead we have turned our attention to *weak-consistency* replication protocols.

Analytical modeling of replication protocols is generally difficult, particularly when many replicas

[†] Supported in part by the Concurrent Systems Project at Hewlett-Packard Laboratories and by a graduate fellowship from the Santa Cruz Operation.

[‡] Supported in part by the National Science Foundation under Grant NSF CCR-9111220.

are involved. Most existing performance evaluation has generally used simulation [11, 7]. We adopted simulation – both discrete event and Monte Carlo – as the appropriate method for evaluating our wide-area architecture.

When weak-consistency replication protocols are used updates are first delivered to one site, then propagated asynchronously to others. The value a server returns to a client read request depends on whether that server has observed the update yet. Eventually, every server will observe the update. This allow updates to be sent using bulk transfer protocols, which provide the best efficiency on high-bandwidth high-latency networks. It also allows the replication protocol to mask many failures, obviating the need for explicit recovery protocols in many cases. Several existing information systems, such as Usenet [13] and the Xerox Grapevine system [14], use similar techniques.

1.1 The Internet environment

The Internet has a high latency required in sending messages, which can affect the response time of an application, and communication unreliability, which may require robust communication protocols. For example, two hosts on an Ethernet can exchange a pair of datagrams in a few milliseconds, while two hosts on the same continent may require 50–200 milliseconds. Packet loss rates of 40% are common, and can go much higher [4]. The Internet has many single points of failure, and at any given time it is usually partitioned into several non-communicating networks. Mobile systems can be disconnected from the network for a long time, or may be "semi-connected" by an expensive low-bandwidth connection. This is a difficult environment for building distributed applications.

Despite this, users expect a service to behave as if it were provided on a local system. The response time of a wide-area application should not be much longer than that of a local one. Further, users expect to use the service as long as their local systems are functioning.

We assume that server processes have access to pseudo-stable storage such as magnetic disk that will not be affected by a system crash. Sites also have loosely synchronized clocks. Sites and processes fail by crashing; that is, when they fail they do not send invalid messages to other processes and they do not corrupt stable storage. Processes can temporarily fail and then recover. Sites have two failure modes: temporary recoverable failures, and permanent removal from service. the network is sufficiently reliable that any two processes can eventually exchange messages, but it need never be free of partitions. *Semi-partitions* are possible, where only a low-bandwidth connection is available.

2 Protocol description

Replicated data can be implemented as a group of replica processes that communicate through a *group communication protocol*. The group communication protocol generally provides a *multicast* service that sends a message from one process to all other processes in the group. The protocol also determines the *consistency* of each replica by controlling the order messages are sent among processes. Weak consistency protocols guarantee that messages are delivered to all members but do not guarantee when.

2.1 Kinds of consistency

The service provided by a process depends on the messages it has received. The level of consistency provided by a replication protocol depends on the consistency guarantees provided by the communication protocol. This includes guarantees on *message delivery, time of delivery,* and *delivery ordering.* In general, strong guarantees require multiphase synchronous protocols while weaker guarantees allow efficient asynchronous protocols.

Messages can either be delivered *reliably,* in which case arrival is guaranteed, or with *best effort,* meaning the system will make an attempt to deliver the message but delivery is not guaranteed. The communication protocol can deliver messages *synchronously,* within a *bounded* time, or *eventually* in a finite but unbounded time.

Messages will be delivered to processes in some order, perhaps different from the order in which they are received. There are several possible orders, including total, causal [9, 8], and bound inconsistency [12, 2]. Weaker orderings include a *per-process* or *FIFO channel* ordering, where the messages from any particular process are delivered in order, but the streams of messages from different processes may be interleaved arbitrarily.

The weak consistency protocols we have developed provide reliable delivery, and can be modified to produce several delivery orderings, but only guarantee eventual message delivery. In particular, all processes are guaranteed to agree in finite but unbounded time if no further messages are sent.

2.2 Timestamped anti-entropy

We have developed a new group communication protocol that provides reliable, eventual delivery, called *timestamped anti-entropy* [5]. The protocol is fault tolerant so messages will be delivered to every process in

the group even if processes temporarily fail or are disconnected from the network. We have also developed a related group membership mechanism that handles adding and removing processes from the replica group [6]. The trade-offs are that the protocol may have to delay message delivery (it is a blocking protocol), that replicas must maintain logs on disk that are not compromised by failure and recovery, and that timestamp information must be appended to every message.

Replicas store messages in a log on stable storage, along with summary information. From time to time each replica selects a partner, and the two exchange messages in an *anti-entropy session.* The replicas can identify what messages the other has not yet received by comparing summary information. When the exchange is complete, the replicas deliver some of the messages from their logs to the database.

The summary information includes a summary timestamp vector, summarizing the messages the replica has received, and an acknowledgment vector, summarizing the messages every other replica has received and acknowledged. Acknowledgments are used to determine when it is safe to remove messages from the log.

There are several variations on the basic timestamped anti-entropy protocol. The timestamp vectors and message timestamps can be used to order messages before they are delivered from the log. Anti-entropy sessions can be augmented by a best-effort multicast to speed propagation, and replicas can use different policies to select partners.

Best-effort multicast can be combined with anti-entropy to spread information rapidly. When a replica originates a message, it can multicast it to other replicas. Some of them will not receive the multicast, either because the network dropped the message or because the replica was temporarily unavailable. These sites will receive the message later when they conduct an anti-entropy session with another site that has received the message. This can speed dissemination when message delivery order is not important.

There are several different policies that replica processes can follow when selecting anti-entropy partners. *Random* selection is the simplest. However, other choices might improve the rate of update propagation or minimize long-distance communication. *Oldest-known* partner selection always selects the partner from which the replica has not received an update in the longest time, in the hope that this site will have the most updates that the replica has not yet observed. *Oldest-biased* tends to bias its selection for out-of-date replicas, but occasionally chooses others. *Distance-biased* partner selection attempts to avoid long-distance communication as much as possible by weighting the chances of randomly selecting a partner based on its distance. Distance biasing was studied as an optimization for Grapevine [3]. A deterministic scheme the

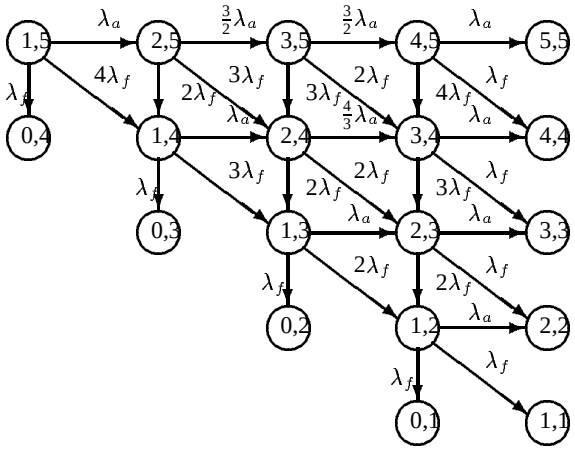

FIGURE 1: Model of failure for five processes. This model only includes permanent failure; temporary failure and recovery would add an additional dimension of states.

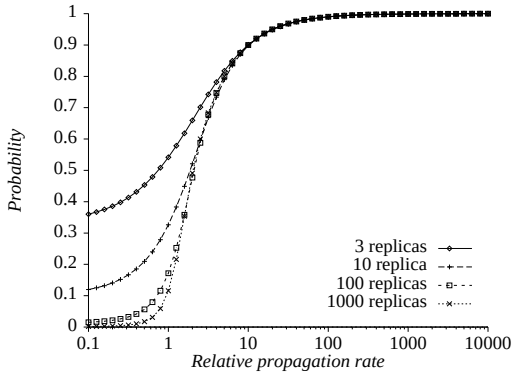

FIGURE 2: Probability of successfully delivering a message to all sites. The relative propagation rate ρ is the ratio of the anti-entropy rate to the permanent site failure rate.

builds a schedule based on *latin squares* [1] is known to be optimal when there are no failures. Fixed-topology schemes, where the replicas are arranged into meshes, trees, or rings are also possible. The performance effects of different selection policies are reported in Section 4.

3 Fault tolerance

We first consider how often site failures would prevent a weak-consistency communication protocol from delivering a message to all replicas.

3.1 Monte Carlo simulation

We modeled information loss using a state transition system like that shown in Figure 1. Each state is labeled with a pair $\langle m, f \rangle$, where m is the number of functioning replicas that have observed a message, and f is the total number of functioning replicas. The system starts with one replica having observed a message out of n possible (5 in the example). The system can then either propagate the information using anti-entropy, or a replica can be removed from service.

We treat anti-entropy as a Poisson process with rate λ_a. The rate of *useful* anti-entropy sessions, where a replica that has an update contacts one that does not, is a function of m and f and the partner selection

policy. In particular, f replicas will be initiating anti-entropy sessions. If replicas choose their partners randomly, each replica that has observed the update has a chance $(f-m)/(f-1)$ of contacting a replica that has not yet observed the update. Since anti-entropy is a Poisson process, the rate of useful anti-entropy sessions is

$$f\frac{f-m}{f-1}\lambda_a.$$

For this evaluation we treated removal from service as a Poisson process with rate λ_f.

Since removal from service is a permanent event, the state transition graph is acyclic, with $O(n^2)$ states in the number of replicas. The probability p_i of reaching each state i can be computed using a sequential walk of the states. The probability density functions $p_i(t)$ of the time at which the system enters each state can be derived analytically or numerically. The analytic solution for $p_i(t)$ can be found by convolving the entry-time distribution $p_j(t)$ for each predecessor state j with the probability density of the time required for the transition from j to i.

We found that systems of hundreds of replicas were too large to solve analytically or numerically, so we used Monte Carlo simulation instead. Since we assumed that all transitions followed a Poisson distribution, the simulator was simple and efficient. We were able to obtain good results for systems of 500 replicas in under two hours on a Sun SparcStation 2 workstation.

3.2 Results

Figure 2 shows the probability of successful delivery for different numbers of replicas, accounting only for permanent failure. The probability is a function of $\rho = \lambda_a/\lambda_f$, the ratio of the anti-entropy rate to the permanent site failure rate. Sites generally are removed from service after several years of service, and unscheduled removals are rare. The anti-entropy rate is likely to be many thousands of times higher than the permanent failure rate. As a result, we expect that there will be no messages lost because of removal from service in a typical system.

Some implementations may buffer messages on volatile storage before copying them to stable storage. This is the default behavior of several operating systems, including Unix. These implementations will lose the information in volatile storage when a replica temporarily fails and recovers.

Volatile storage complicates the model. States must be labeled with four values: the number of functioning replicas that have not observed a message, the number that have written it to volatile store, the

number that have written it to disk, and the number that have temporarily failed. The state transitions are complex and the solution is impractical for realistic numbers of replicas. We performed a Monte Carlo evaluation of the system to obtain estimates of fault tolerance.

The effect of volatile storage can be bounded by considering the probability that a failure will occur while there are unstable messages. Assume that temporary failure is a Poisson process with rate λ_t and that volatile data is flushed to stable storage every s time units. The probability that a failure occurs before writeback is

$$p = \frac{-2e^{-s\lambda_t} + s^2\lambda_t^2 - 2s\lambda_t + 2}{2s\lambda_t^2}.$$

For a typical value of $s = 30$ seconds and $1/\lambda_t = 15$ days, p is so close to zero as to be negligible.

4 Consistency and delay

Weak consistency protocols allow replicas to contain out-of-date information. There are two related measures of this effect—one concerning the propagation of an update, the other concerning the consistency of replicated values. The time required to propagate an update from one replica to others shows how quickly information will be made available to clients. The likelihood of encountering out-of-date information, and the age of this information, aggregates the effects of several updates.

4.1 Discrete event modeling

We constructed two discrete event simulation models of the timestamped anti-entropy protocol: one to measure propagation delay and the other to measure information age.

The propagation delay simulator measured the time required for an update message, entered at time zero, and its acknowledgments to propagate to all available replicas. It used two events: anti-entropy propagation and permanent site failure. The simulator could be parameterized to use different partner selection policies. The simulator was run until either the 95% confidence intervals were less than 5%, or 10 000 updates had been processed. In practice 95% confidence intervals were generally between 1 and 2%.

The information age simulator was more complex. It used five events: one each to start and stop the simulation, one to send update messages, one to perform anti-entropy, and one to read data from a replica. The simulation was allowed to run for 1 000 time units so it would reach steady state. The simulation ended

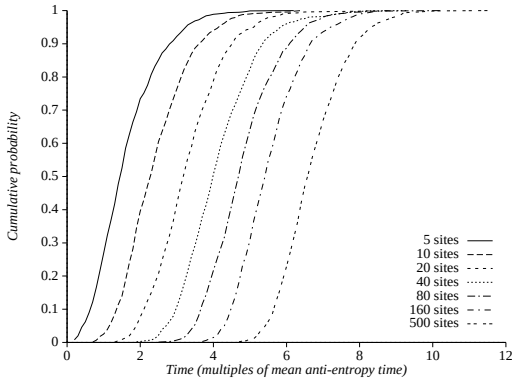

FIGURE 3: Cumulative probability distribution for propagating a message to all replicas. Measured for random partner selection with $\rho = 1000$.

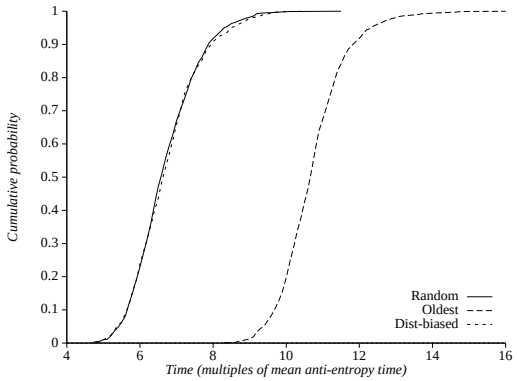

FIGURE 4: Effect of partner selection policy on propagation delay. Measured for 500 sites with $\rho = 1000$.

at 50 000 time units. In no case did the 95% confidence interval width exceed 4% of the mean. Read, write, and anti-entropy events were modeled as Poisson processes. The simulator included several partner selection protocols and an optional unreliable multicast on writes.

The simulator maintained two data structures for each replica: the anti-entropy summary vector and a data version number. It also maintained a global data version counter. Write events incremented the global version counter and copied that value to the version number for some replica. If an unreliable multicast was being used, the version number would be copied to other replicas if a simulated unreliable datagram was received. Anti-entropy events propagated version numbers between replicas, as well as updating the summary vectors.

Read events were used to collect measures of the expected age of data and the probability of finding old data. A replica was selected at random, and the version number for that replica was compared to the global version. The difference showed how many updates the replica had yet to receive.

4.2 Propagation delay

The time required to propagate a message from one replica to others was the first measure we investigated. If information is propagated quickly, clients using different replicas will not often observe different information, and loss of an update from site failure will be unlikely. The size of the message log is related to this measure, since messages are removed from the log when acknowledgments have been received from every replica.

Figure 3 shows the cumulative probability over time that an update has been received by all replicas. Time is measured as multiples of the mean interval at which replicas initiate anti-entropy events. Anti-entropy was assumed to occur 1 000 times as often as permanent site failure—as we have noted, a low estimate. The simulations in this graph use random partner selection with no initial multicast.

This figure also shows that our weak-consistency protocols scale well to large numbers of sites. The propagation delay increases roughly as the logarithm of the number of sites.

We also considered the effect of different partner selection policies on the speed of propagation. Figure 4 shows the distribution for different policies, using 500 sites and without initial multicast. Random and distance-biased selection are nearly nearly identical, while the oldest-first policy is generally somewhat slower than the others. The speed of acknowledgment propagation is also important. We have found that acknowledgment delay scales as well as propagation delay, and that the oldest-first policy is slower than the other two.

4.3 Age of information

While propagation delay measures the performance of one update, this measure shows the consistency of concurrent updates on a single data item. We took two measures: the probability that a replica would return old information, regardless of its age, and the expected age of information whether it was current or not.

The age of a database entry depends on the ratio of the anti-entropy rate to the update rate for that entry. Many wide-area services have extremely low update rates; some services write new entries and never change them. A low update rate means that anti-entropy has a better chance of propagating an update before another update enters the system. In the Domain Name Service [10], a host name or address rarely changes more than once every few months. In other databases new entries are added, corrected quickly, then remain stable. We expect the update rate for a single database entry to be about a thousand times lower than the anti-entropy rate. Most of the graphs in this section were generated using a mean time-to-update of 1 000 time units; the maximum anti-entropy rate investigated was only 200 times greater, giving a mean time-to-anti-entropy of five.

Figure 5 shows the likelihood of getting out-of-date information, while Figure 6 shows the expected age of that information. Clearly, adding an unreliable multicast on write significantly improves both measures. We have found that the message success probability is the most important influence on information age in large groups of replicas. For small numbers of sites, increasing the anti-entropy rate dramatically improves

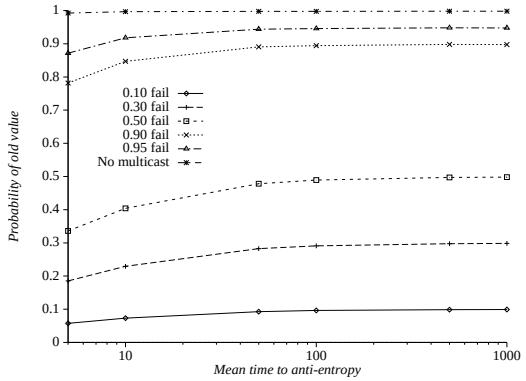

FIGURE 5: Probability of getting old value as anti-entropy rate varies, for 500 sites. Mean time-to-write 1 000; random partner selection.

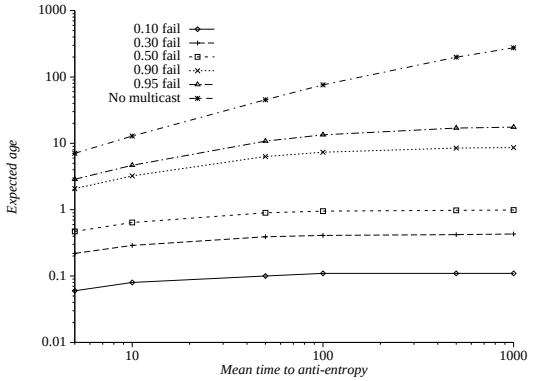

FIGURE 6: Expected data age as anti-entropy rate varies, for 500 sites. Mean time-to-write 1 000; random partner selection.

both the probability of getting up-to-date information and the expected age.

Figures 7 and 8 show how consistency depends on the number of sites. For these simulations the anti-entropy rate was fixed at 100 times that of writes. We expect this value might be typical for a database entry soon after it is entered, when updates are most likely. Later updates will be less frequent and the ratio will increase, improving the consistency. Once again an unreliable multicast provides considerable improvement.

We also investigated the effect of partner selection policy on information age. Since random and distance-biased selection showed identical propagation rates, it is no surprise that they have the same expected age. Oldest-first selection provides slightly older information, consistent with a slower propagation rate.

5 Conclusions

Scalability and fault tolerance are two of our goals for a wide-area distributed service architecture. Voting protocols cannot meet this goals in a wide-area network, so we have adopted weak-consistency replication protocols. These features are essential for making a system usable for mobile computing systems.

Simulation has proven a useful mechanism for investigating these goals. It would have been difficult to build and measure a system on the Internet with hundreds of replicas, particularly if we wanted to evaluate the protocols under specific failure conditions.

We have found that the timestamped anti-entropy protocol scales well to large numbers of replicas. We

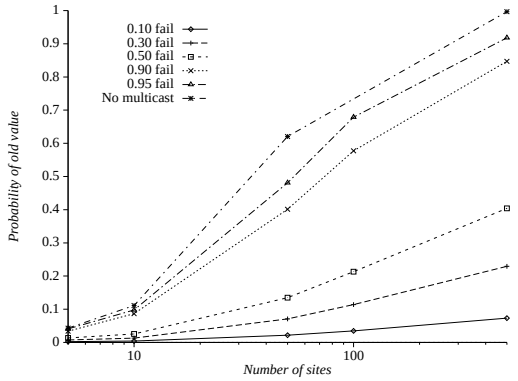

FIGURE 7: Probability of getting old value as the number of sites varies, with anti-entropy occurring 100 times as often as writes. Random partner selection.

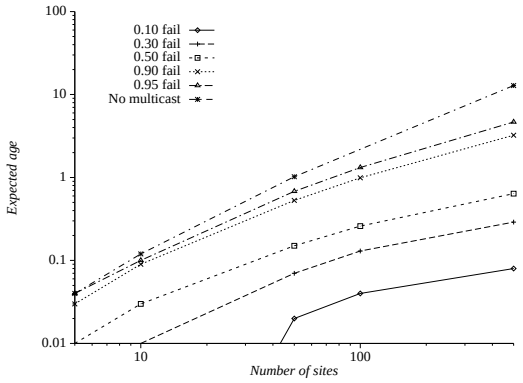

FIGURE 8: Expected data age as the number of sites varies, with anti-entropy occurring 100 times as often as writes.

simulated groups of up to 500 replicas, and found that the time required to propagate an update from one replica to all other replicas increases as the log of the size of the group.

Fault tolerance is the ability of a system to provide correct service even when some parts fail. The timestamped anti-entropy protocol avoids synchronous communication, instead communicating between pairs of replicas. We found that most partner selection policies are robust in the face of site and network failure.

The reliable delivery guarantee is not met in one important case: when a process permanently fails and loses data. No weak consistency communication scheme can be free from this, since a window of vulnerability exists while the data is being sent to other processes. Our simulation results show that the duration is insignificant in practice..

This can be contrasted with consistent replication protocols, such as voting, that require synchronous communication with a majority of replicas. Simulations of voting protocols [11, 7] shows they provide good availability and performance with small numbers of replicas; however, they are not practical for hundreds or thousands of replicas. Consistent replicas cannot continue to function when disconnected from other replicas, so they are not useful for mobile computing systems.

References

[1] N. Alon, A. Barak, and U. Manber, "On disseminating information reliably without broadcasting," in *Proceedings of 7th International Conference on Distributed Computing Systems*, pp. 74–81, 1987.

[2] D. Barbará and H. Garcia-Molina, "The case for controlled inconsistency in replicated data," in *Proceedings of the Workshop on the Management of Replicated Data*, pp. 35–8, November 1990.

[3] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," *Operating Systems Review*, vol. 22, pp. 8–32, January 1988.

[4] R. A. Golding, "Accessing replicated data in a large-scale distributed system," Master's thesis, Computer and Information Sciences Board, University of California at Santa Cruz, June 1991. Published as Tech. Rep. UCSC–CRL–91–18.

[5] R. A. Golding, "The timestamped anti-entropy weak-consistency group communication protocol," Tech. Rep. UCSC–CRL–92–29, Computer and Information Sciences Board, University of California at Santa Cruz, July 1992.

[6] R. A. Golding and K. Taylor, "Group membership in the epidemic style," Tech. Rep. UCSC–CRL–92–13, Computer and Information Sciences Board, University of California at Santa Cruz, April 1992.

[7] S. Jajodia and D. Mutchler, "Dynamic voting," in *Proceedings SIGMOD International Conference on Data Management*, pp. 227–238, ACM, 1987.

[8] R. Ladin, B. Liskov, and L. Shrira, "Lazy replication: exploiting the semantics of distributed services," *Operating Systems Review*, vol. 25, pp. 49–55, January 1991.

[9] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, no. 7, pp. 558–65, 1978.

[10] P. Mockapetris, "Domain names – concepts and facilities," Tech. Rep. RFC 1034, ARPA Network Working Group, November 1987.

[11] J.-F. Pâris, "Voting with witnesses: A consistency scheme for replicated files," in *Proceedings 6th International Conference on Distributed Computing Systems*, (Cambridge), pp. 606–612, IEEE, 1986.

[12] C. Pu and A. Leff, "Replica control in distributed systems: an asynchronous approach," Tech. Rep. CUCS–053–090, Department of Computer Science, Columbia University, January 1991.

[13] J. S. Quarterman and J. C. Hoskins, "Notable computer networks," *Communications of the ACM*, vol. 29, pp. 932–71, October 1986.

[14] M. D. Schroeder, A. D. Birrell, and R. M. Needham, "Experience with Grapevine: the growth of a distributed system," *ACM Transactions on Computer Systems*, vol. 2, pp. 3–23, February 1984.