Modeling replica divergence in a weak-consistency protocol for global-scale distributed data bases

Richard A. Golding[†]
Vrije Universiteit, Amsterdam, The Netherlands

Darrell D. E. Long[‡]
University of California, Santa Cruz

UCSC–CRL–93–09

Concurrent Systems Laboratory
Computer and Information Sciences
University of California, Santa Cruz
Santa Cruz, CA 95064

Distributed database systems for wide-area networks must scale to very large numbers of replicas in order to provide acceptable availability and response time. Weak-consistency replication protocols, such as the *timestamped anti-entropy* (TSAE) protocol we have developed, allow a database to scale to hundreds or thousands of replicas. The TSAE protocol allows updates to be processed by a single replica, then propagated from one replica to another in the background, causing replicas to temporarily diverge. The divergence is resolved in a short time and is resolved correctly even in the presence of temporary replica failure and network partition. We present a detailed analysis of the update propagation latency and the divergence between replicas caused by this protocol.

[†]Supported in part while at the University of California 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 and by the Office of Naval Research under Grant N00014–92–J–1807.

1 Introduction

Wide-area information systems are becoming an important resource in many fields. Given the vast scale of the Internet, such systems provide unique challenges that cannot be met by traditional distributed data base designs. Some of these systems must scale to millions of users located in every part of the world, while providing good availability and interactive response. At the same time, the system must respond gracefully to host and network failure, to the long latencies required to communicate between distant systems, and to the presence of mobile computer systems.

We have developed an architecture for constructing wide-area services that uses *group communication* to implement a replicated service [Golding92c]. A number of *replicas* or servers form a group, and coordinate their actions using a group communication mechanism. The mechanism can be tailored to the specific needs of the service by combining the appropriate low-level communication, group management, and message management modules, which can provide guarantees ranging from traditional single-copy serializability using a strongly-consistent protocol such as quorum consensus [Gifford79, Thomas79], to a weak-consistency mechanism using the *timestamped anti-entropy (TSAE)* protocol that we have developed.

Weak-consistency protocols, sometimes called *epidemic* replication protocols, provide eventual message delivery. A database update is sent as a message to one replica. The message is propagated from one replica to another in the background, eventually reaching every replica in the group.

This approach provides a good solution for building a replicated service on a wide-area network. Background propagation avoids synchronous interactions between more than two replicas at a time, and clients need only interact with one nearby replica to access the service. A service can place replicas near client sites, spreading processing load over many servers, decreasing the latency clients require to communicate with the service, and minimizing the long-distance network traffic between client and service. Consistent replication protocols require a synchronous interaction between a client and multiple replicas, requiring a client to wait for communication with distant replicas. The latency of contacting hundreds of replicas synchronously over the Internet, and the message traffic required, is unacceptable.

Delayed propagation makes such protocols extremely robust, because propagation can wait for faults to be repaired without compromising delivery guarantees. The replicas are allowed to diverge, then are reconciled when the fault is repaired. This is an advantage for large-scale services on the Internet, which is never without partitions and failed hosts. A consistent replica, which must always preserve single-copy serializability, cannot provide service when disconnected from other replicas and is therefore less available. Replicas can batch messages between them, using bulk transfer protocols at off-peak times.

These same features make weak consistency attractive for mobile computers. If a replica resides on a mobile computer, the service will be accessible even when the computer is disconnected from the network. The replica will diverge from the others until it is reconnected to the network and can exchange updates with other replicas.

Our *timestamped anti-entropy (TSAE)* protocol is a significant improvement over previous weak-consistency protocols. It provides reliable message delivery while ensuring that network traffic is kept to a minimum. It also maintains enough information to support correct truncation of message logs, as well as causal or total message delivery orders.

We have used the TSAE protocol to prototype a large-scale, wide-area distributed bibliographic database management system, called Refdbms [Golding92b]. Several Refdbms databases are replicated at sites in North America and Europe. Other existing information systems, such as Usenet [Quarterman86] and the Xerox Clearinghouse system [Oppen81, Demers88], use other weak-consistency techniques. Our work formalizes and improves on these *ad hoc* approaches, providing a single framework for analyzing, comparing, and implementing them [Golding92c].

Clients using a weakly-consistent service can observe out-of-date or inconsistent information, unlike clients of a service that provides single-copy serializability. We have investigated two measures of the degree to which clients will observe out-of-date results – one concerning the propagation of a single message, the other concerning the divergence between replicas. The time required to propagate a message from one replica to others shows how quickly information will be made available to clients; this is discussed in Section 3. The likelihood that a replica is out-of-date with respect to other replicas, and the difference between them, aggregates the effects of several messages. We discuss this measure in Section 4. We have evaluated the fault tolerance of message delivery and of the group management mechanism, and the network traffic imposed by the protocol, as we have reported elsewhere [Golding93, Golding92b].

In the remainder of this section we will justify why weak-consistency protocols are necessary for the large-scale wide area systems that are currently being built. In Section 2 we summarize the TSAE protocol.

1.1 The wide-area networking environment

Consistent replication protocols are unsuited for wide-area applications because of the latency, unreliability, and scale of those networks. Latency affects the response time of the application, and can vary from a few milliseconds, for two hosts connected by an Ethernet, to several hundred milliseconds for hosts on different continents communicating through the Internet. Packet loss rates often reach 40%, and can go higher [Golding92a]. Further, the Internet has many points that, on failure, partition the network, and at any given time it is usually partitioned into several non-communicating networks. Further, in January 1993 the Internet included more than 1.3 million hosts with an estimated 12 million users [Lottor93]. This has led to query loads on some services exceeding the capacity of a single server and the network links that support it [Emtage92]. Any replication protocol that requires interactive communication with many replicas will not work in this environment.

The introduction of mobile computers exacerbates this problem further. These systems spend most of their time disconnected from other systems, or perhaps connected by an expensive or low-bandwidth link. Services that are to support such systems must allow clients that are disconnected to continue to operate, without communicating with outside servers. This can be accomplished by placing a replica on the mobile system and allowing the copy to diverge from the "correct" value. Weak consistency protocols ensure that this divergence can be reconciled when the mobile system is reconnected to the network. The TSAE protocol allows mobile systems with limited bandwidth to measure how far they have diverged by exchanging a small summary of the state of the replica with another replica.

Despite these restrictions, 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.

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 provides a *multicast* service that sends a message from one replica to all other replicas in the group. The protocol controls the consistency, or divergence, of each replica by controlling the latency, reliability, and order of the messages sent among them.

Strong consistency requirements are impossible to meet in the most general cases, and expensive in others. For example, if there are no bounds on message delivery time it is not possible to guarantee consistency [Turek92]. If replicas can fail in arbitrary ways, providing reliable delivery is equivalent to Byzantine Agreement.

For most applications the Internet can be treated as an unreliable network with bounded communication latency. The hosts on the network approximate synchronous processors that fail by crashing; that is, no processor is infinitely fast or slow; every failure is recovered in a finite time; and when a failure occurs the processor neither sends invalid messages to other processors nor corrupt stable storage. Consensus is theoretically possible under these conditions.

We also assume that replicas have access to pseudo-stable storage such as magnetic disk that will not be affected by a system crash. Replicas, or the hosts on which they run, have loosely synchronized clocks. The network is sufficiently reliable that any two replicas can eventually exchange messages, but it need never be free of partitions. *Semi-partitions* are also possible, where only a low-bandwidth connection is available.

2.1 Kinds of consistency

We view consistency in terms of the messages that are exchanged between replicas. In general, two replicas are consistent at a particular moment if they have received the same set of messages. Unlike some other work on distributed consistency, we reason about consistency using real time that could be measured by an outside observer rather than a virtual time measure.

We have developed a framework for constructing and classifying group communication mechanisms [Golding92c]. In this approach, we classify a mechanism by the *latency* and *reliability* of message delivery, and by the order in which messages are delivered to the service.

The communication protocol can deliver messages *synchronously,* within a *bounded* time, or *eventually* in a finite but unbounded time. Weak-consistency protocols provide eventual delivery, because there is no bound on the interval at which a replica will be able to contact another to propagate a message.

Messages can either be delivered *atomically,* so that either every replica receives the message or none do; *reliably,* in which case the message is delivered to every functioning replica, but failed replicas need not receive it; or with *best effort,* meaning the system will make an attempt to deliver the message but delivery is not guaranteed. Weak consistency protocols provide reliable delivery.

The reliable delivery guarantee differs from atomic delivery in one important case: when the sending replica permanently fails and loses data. No weak consistency communication scheme can provide atomic delivery, since a window of vulnerability exists while the data is being sent to other replicas. In practice the duration of the window is insignificant.

The Internet provides no guarantees on the order messages are received by a replica. The replica can re-order the messages it receives before they are applied to the database. The ordering can be *total,* so that every message is delivered in the same order at every replica; *causal,* so that the ordering may be different at different replicas as long as every ordering respects potential causal relations between messages [Lamport78]; or *unordered,* where the ordering is not coordinated between replicas. Some weak-consistency protocols, such as TSAE, allow replicas to order messages causally or totally.

Several protocols that provide weakly consistent replication have been proposed. The Xerox Clearinghouse [Demers88, Oppen81] name service used three epidemic replication protocols, including best-effort multicast, rumor mongery, and anti-entropy. Of these three, only anti-entropy provided reliable delivery, and it could not support message reordering or provably reliable log purges. The Lazy Replication system [Ladin90] supported causal and total message orderings, including orderings that respected casual relations caused outside the replica group. However, the protocol itself was notably inefficient. The composition of TSAE with a causal message ordering module is equivalent to the Lazy Replication system.

2.2 Timestamped anti-entropy

The TSAE protocol is a new group communication protocol that provides reliable, eventual delivery [Golding92b]. Like other weak-consistency protocols, update messages originate at a single replica and are propagated in the background to others. Unlike most others, TSAE can support total or causal message delivery orders, mobile computer systems, and provably correct purging of message logs. We have developed a compatible group membership mechanism for adding and removing replicas from the group.

When a replica wishes to send a message, it stamps the message with the current time and the identity of the replica, then writes the message to a log. This log is maintained on stable storage, so that it survives temporary crashes. It is organized as a set of sub-logs, each holding messages sent by one replica in the group. The top part of Figure 1 shows an example of the logs at two replicas, A and B, that are part of a group of three replicas.

From time to time, a replica will select another replica, and the two will exchange the contents of their message logs in an *anti-entropy session.* At the end of the session, both replicas have received the same set of messages. Moreover, they have received a continuous sequence of messages from each replica, with no gaps. To see that this is so, assume that the condition holds before a session, as shown in the top of Figure 1. Replica A has a complete set of the four messages it has sent, while B has only the first two. During the session A sends the *entire* set of messages that B does not have. At the end, both replicas have identical sets of messages in their logs.

Each replica maintains *summary* information that the TSAE protocol uses to make message exchange efficient. Each replica maintains a summary timestamp vector, indexed by replica identifier, containing the greatest timestamp it has received from other replicas. In the example in Figure 1, replica A has received messages sent by C with timestamps as great as 4. Replica B, on the other hand, has only received messages from C up to time 2. Since anti-entropy ensures that replicas receive continuous message sequences, having one message in the log implies that all previous ones have been received. When a replica's summary vector holds a timestamp for another replica, every message with a lesser or equal timestamp has been received.

An anti-entropy session using the TSAE protocol begins with two replicas exchanging their summary vectors. Each replica can determine what messages its partner has not yet received by comparing its

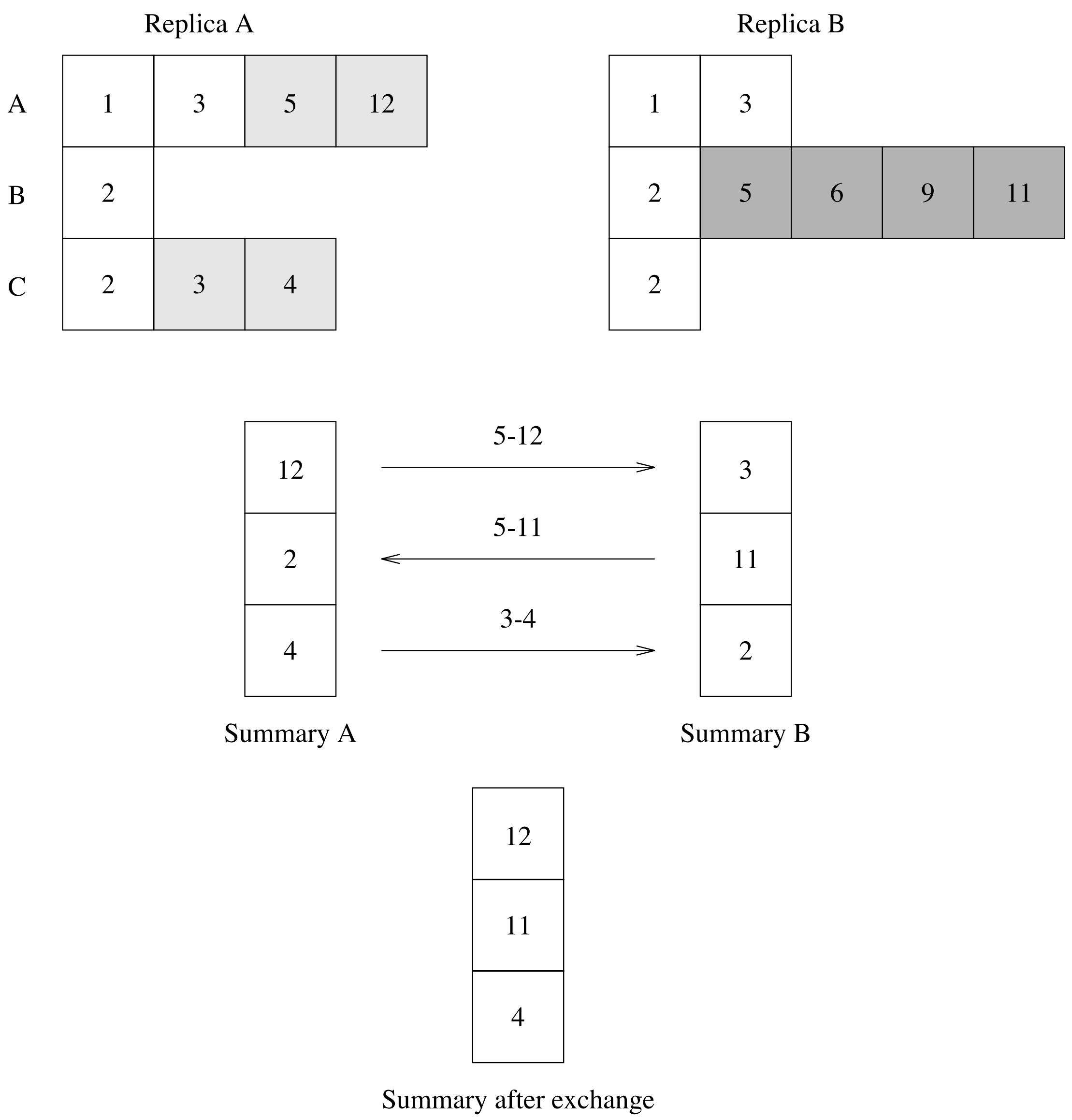

FIGURE 1: An example anti-entropy session. Principals A and B begin with the logs in the top of the figure. They exchange summary vectors, discovering that the shaded messages must be exchanged. After the exchange, they update their summary vectors to the bottom vector.

summary vector to that of its partner. These messages are sent using a reliable stream communication protocol. Once both replicas have received their messages, they can update their summary vector to the elementwise maximum of their original vector and their partner's vector. In Figure 1, replica A can determine that it has received messages 3 and 4 from replica C, but that B has not. This ensures that every message is sent exactly once to each replica.

While the summary vector records the messages that one replica has received, replicas also need information on the messages other replicas have received in order to truncate the message log safely. Rather than use explicit acknowledgments for every message, the TSAE compresses them into another summary vector. At the end of an anti-entropy session, a replica finds the minimum timestamp in its summary vector. The replica has received every message from any sender that has a lesser or equal timestamp. This method makes progress as long as replicas have loosely-synchronized clocks.[1]

Each replica maintains a vector of these acknowledgment timestamps, one for each replica in the group, and exchanges this vector with its partner during anti-entropy sessions. Any message in the log whose timestamp is less than every timestamp in the acknowledgment vector has been received and acknowledged by every replica in the group, and so can be safely removed from the log.

A replica using the TSAE protocol only needs to initiate anti-entropy sessions when there are messages in its log that have not been received and acknowledged by other replicas. This allows a quiet group to create message traffic only when an update has arrived, unlike other weak-consistency protocols.

2.3 Best-effort multicast

While the TSAE protocol is efficient, some applications may perform better if information is propagated more rapidly. One technique is to combine a best-effort multicast with anti-entropy. When a replica sends an update message to the replica group, it can first multicast the message to other replicas, many of which will receive the message. Anti-entropy sessions can later ensure that any replica that missed the multicast – either because the network was unreliable, or because the replica was temporarily unavailable when the multicast occurred – receives the message.

2.4 Partner selection

When a replica selects another replica for an anti-entropy session, it can use one of several *partner selection policies*. The choice of policy affects message delivery latency and hence the degree of consistency between replicas, and the amount of network traffic caused by the protocol. Table 1 lists eight policies we have examined.

The policies can be divided into three classes: random, deterministic, and topological. Random policies assign a probability to each replica, then randomly select a partner for each session. The deterministic policies use a fixed rule to determine the replica to select as partner, possibly using some extra state such as a sequence counter. Topological policies organize the replicas into some fixed graph structure such as a ring or a mesh, and propagate messages along edges in the graph.

The **uniform** policy assigns every replica an equal probability of being selected as partner. Uniform selection can lead to overloaded network links in an internetwork where the physical topology is less connected than the logical.

Demers et al. compared uniform to **distance-biased** selection [Demers88]. Their study found that biasing partner selection by distance could reduce traffic on critical intercontinental links by more than an order of magnitude. Selection can also be biased by the cost of communication, perhaps measured in terms of latency, or monetary cost of using a communication link.

[1]We have also developed a similar protocol that requires $O(n^2)$ state per replica rather than $O(n)$, but allows unsynchronized clocks. This alternate protocol was discovered independently by Agrawal and Malpani [Agrawal91].

TABLE 1: Partner selection policies.

Random policies:	
Uniform	Every other replica has an equal probability of being randomly selected.
Distance-biased	Nearby replicas have a greater probability than more distant replicas of being randomly selected.
Oldest-biased	The probability of selecting a replica is proportional to the age of its entry in the summary vector.
Deterministic policies:	
Oldest-first	Always selects the replica n with the oldest value summary$[n]$.
Latin squares	Builds a deterministic schedule guaranteed to propagate messages in $\Theta(\log n)$ rounds.
Topological policies:	
Ring	Organizes the replicas into a ring.
Binary tree	Replicas are organized into a binary tree, and messages are propagated randomly along the arcs in the tree.
Mesh	Organizes the replicas into a two-dimensional rectangular mesh.

Alon et al. [Alon87] proposed the **latin square** policy, which guarantees that a message is received by all replicas in $O(\log n)$ time (assuming no replica failure). A latin square is an $N \times N$ matrix of N entries, where every row and column includes every entry once. The policy builds a communication schedule by constructing a random latin square, where the columns in the matrix are the schedules for each replica. A replica cycles through its schedule, contacting replicas in the order given, and skipping over itself. It is not evident how to take advantage of topological information in this approach. It is also not clear how to extend it for dynamically changing groups without performing a consistent computation to build new schedules, since each replica must build and follow the same schedule for selecting partners.

The **oldest-biased** and **oldest-first** policies attempt to produce the same effect as **latin squares** without computing a global schedule. **Oldest-biased** randomly selects a partner with probability proportional to the age of its entry in the summary vector. **Oldest-first** always selects the oldest entry, breaking ties by selecting the "closer" entry if it can be determined.

The topological policies, including **ring, binary tree,** and **mesh,** organize replicas into a regular graph. Messages are propagated along edges in the graph. A topological policy can work well when its structure can be mapped onto the structure of the network.

3 Message latency

The TSAE protocol only guarantees eventual delivery, but in practice messages propagate to every replica rapidly. 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.

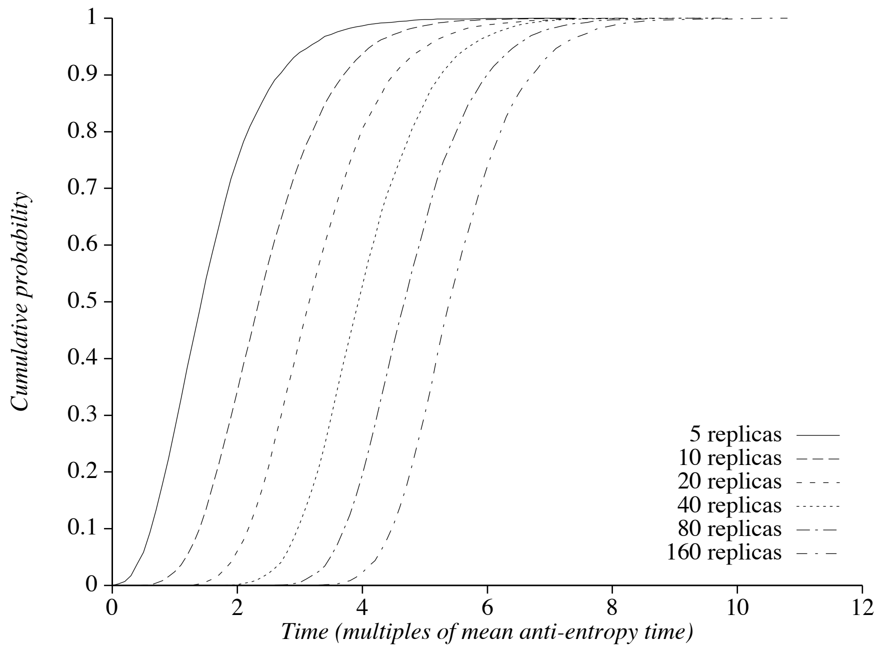

FIGURE 2: Cumulative probability distribution for propagating a message to all replicas. Measured for **uniform** partner selection.

We constructed a discrete event simulation model of the TSAE protocol to measure propagation latency. The latency simulator measured the time required for an update message, entered at time zero, and its acknowledgments to propagate to all available replicas. The time required to send a message from one replica to another was assumed to be negligible compared to the time between anti-entropy sessions. The simulator could be parameterized to use different partner selection policies and numbers of sites. The simulator was run until either the 95% confidence intervals on the mean message and acknowledgment latency were less than 5%, or 10 000 updates had been processed. In practice 95% confidence intervals were generally between 1 and 2%.

The simulation modeled only the TSAE protocol, and did not consider the effect of combining TSAE with a best-effort multicast. Therefore the results in this section represent worst-cast behavior that would be improved if a multicast were added.

Figure 2 shows the cumulative probability over time that a message has been received by all replicas for varying numbers of replicas. Time is measured as multiples of the mean interval at which replicas initiate anti-entropy events. The simulations in this graph use **uniform** partner selection. The time required to propagate a message appears to scale well with the number of sites.

The time required to propagate message acknowledgments everywhere is an important measure, because it determines how quickly messages can be purged from the message log. Figure 3 shows the latency required from the time a message is sent to the time acknowledgments are received by every replica from every replica. Acknowledgment latency required appears to scale as well as propagation latency.

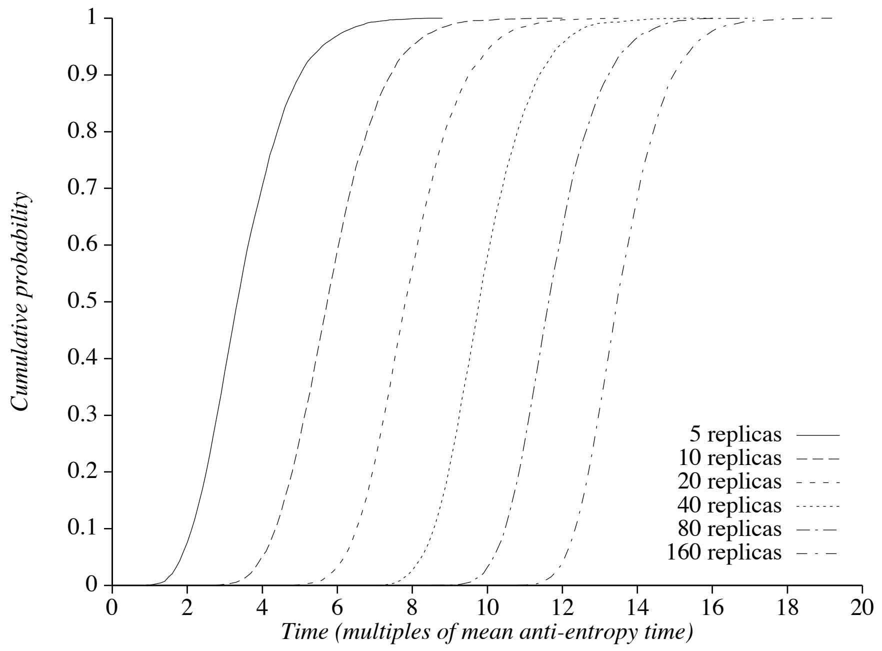

FIGURE 3: Cumulative probability distribution for receiving an acknowledgment from all replicas. Measured for **uniform** partner selection.

The partner selection policy affects the speed of message propagation. Figure 4 shows the mean time required to propagate a message to every replica for several policies as the number of sites increases. The **uniform, latin squares,** and **distance-biased** policies give essentially identical performance. **Age-biased** appears to provide slightly better performance, which would appear to contradict the claim by Alon that the **latin squares** policy is fastest [Alon87]. We believe the difference arises from a slight difference in implementation: Alon's implementation requires that every replica propagate messages in well-defined rounds, while this simulation allows propagation to occur at random intervals. This may mitigate some of the benefit derived from Alon's **latin squares** policy. The policies that simulate a fixed topology – **ring, mesh,** or **binary tree** – have the worst performance and scaling. The results for acknowledgment time are similar.

These results indicate that simple random policies, such as uniform selection or age biasing, perform quite well. We used **uniform** partner selection in the Refdbms system.

4 Replica consistency

Consistency measures the difference between replicas. A small difference means that the replicas are nearly consistent, and clients performing queries at different replicas will observe nearly the same value. Consistency can either be measured over the database as a whole, or for an individual entry. It can also be measured simply as the probability that a replica is out-of-date, or as the number of updates the replica has yet to receive.

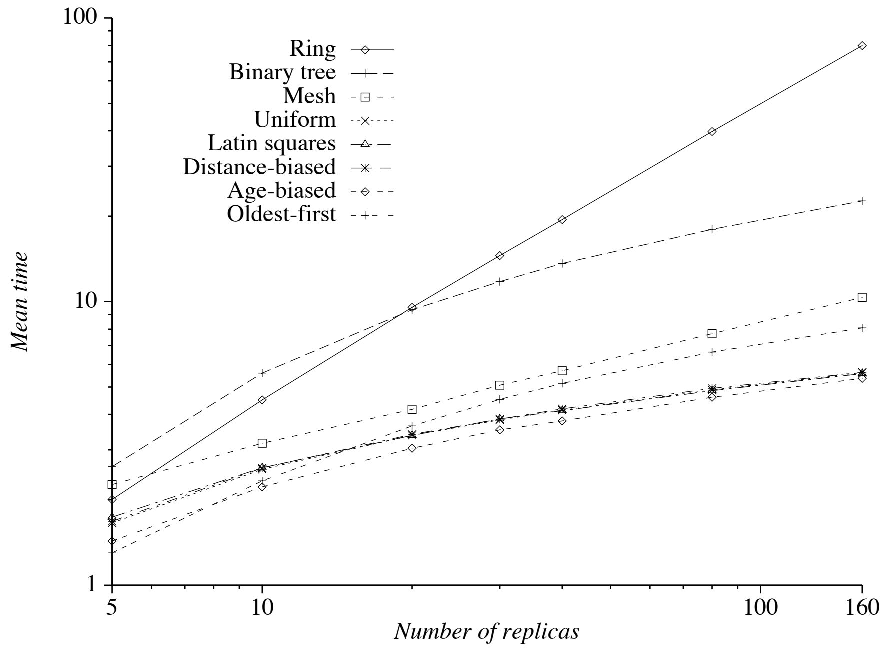

FIGURE 4: Effect of partner selection policy on scaling of propagation time.

We have chosen to measure consistency as the number of updates a replica has missed over the entire database, calling this the *age* of the replica's copy. We expect that most widely-shared databases will be very large, containing thousands to millions of entries. A typical Refdbms database holds several thousand bibliographic references. We also expect that these databases will often be used to search for information, giving rise to queries that potentially consider a significant fraction of the database in computing a result. For example, in Refdbms we often find users asking for all references on a particular subject. Consistency aggregated over the entire database is more useful than consistency of individual items for measuring the accuracy that this sort of query will observe.

The age of a replica's state depends on the ratio of the anti-entropy rate to the update rate for the state. 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 [Mockapetris87], a particular host name or address rarely changes more than once every few months. In systems like Refdbms, new entries are added, corrected quickly, then remain stable. We expect the update rate for most wide-area services to be much 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 mean time-to-anti-entropy varied from 5 to 1 000 time units.

Once again we used a discrete event simulation to model the TSAE protocol. The simulator used five events: to start and stop the simulation; to send a message; to perform anti-entropy; and to sample the state of a replica. The simulation was first allowed to run for 1 000 time units so it would reach steady state, then measurements began. The simulation ended at 50 000 time units. Read, update, and anti-entropy events

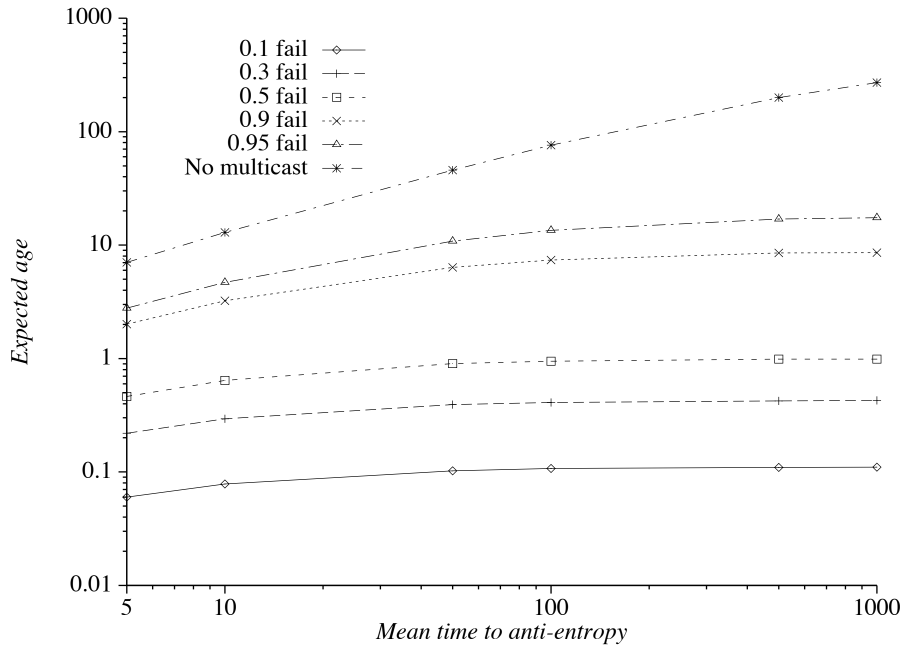

FIGURE 5: Expected replica age as anti-entropy rate varies, for 500 replicas. Mean time-to-update 1 000; **uniform** partner selection. Anti-entropy was combined with a best-effort multicast, for which the message failure rate varied.

were modeled as Poisson processes with parameterizable rates. These rates were measured per replica. The simulator included different partner selection protocols and an optional unreliable multicast on updates.

The simulator maintained two data structures for each replica: the anti-entropy summary vector and a message number. It also maintained a global message counter. When a message was sent, the global counter was incremented and the sender's message number was assigned that value. If an unreliable multicast was being used, the message number was copied to other replicas if the datagram was received. Anti-entropy events propagated message numbers between replicas, as well as updating the replicas' summary vectors.

Sampling events collected the expected age of a replica's data and the probability of finding old data. A replica was selected at random, and the message number for that replica was compared to the global counter. The difference showed how many messages the replica had yet to receive.

Figure 5 shows the expected age of the value held by a replica. Clearly, adding an unreliable multicast on update significantly improves this measure. The message success probability is the most important influence on replica age in large groups of replicas. For small numbers of replicas, increasing the anti-entropy rate dramatically improves both the probability of getting up-to-date information and the expected age.

Consistency also depends on the number of replicas, as shown in Figure 6. For these simulations the anti-entropy rate was fixed at 100 times that of update. This value might be typical for a Refdbms entry soon after it is entered, when corrections 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.

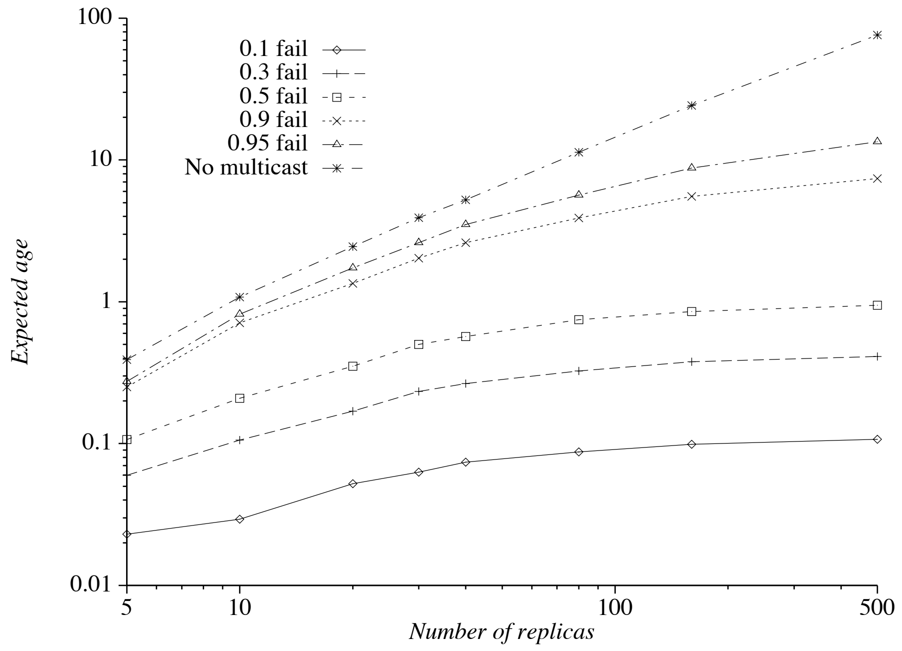

FIGURE 6: Expected replica age as the number of replicas varies, with anti-entropy occurring 100 times as often as updates. Uses **uniform** partner selection. Also shows the effect of varying message failure rates in a best-effort multicast.

We also investigated the effect of partner selection policy on replica age, as shown in Figure 7. The results show that expected age is related to propagation time, since the policies are ranked in exactly the same order as in Figure 4, which shows the mean propagation time for the different policies. The topological policies (**Ring, binary tree, and mesh**) propagate more slowly, and give a greater expected age, than other policies. The other policies are nearly equal, though **oldest-first** has a slight advantage.

5 Conclusions

Wide-area distributed database systems must scale to millions of users, and must operate correctly on an unreliable network. A replicated database with hundreds or thousands of replicas can meet availability and query performance goals if it uses weak-consistency replication protocols. Many widely-shared databases, including name services and bibliographic databases, are not concerned with strict consistency.

The timestamped anti-entropy (TSAE) weak-consistency protocol avoids synchronous communication between replicas and clients, instead propagating updates in the background. When a replica is partitioned from the rest of the network, it can continue to provide service and will receive updated information once it reconnects. Likewise, no special protocols are required for recovery from temporary failure. TSAE scales well, requiring $O(n)$ state per replica for a simple implementation.

This can be contrasted with consistent replication protocols, such as voting, that require synchronous communication with a majority of replicas. While consistent protocols can provide good availability and performance with small numbers of replicas, they are not practical for global-scale networks. Consistent

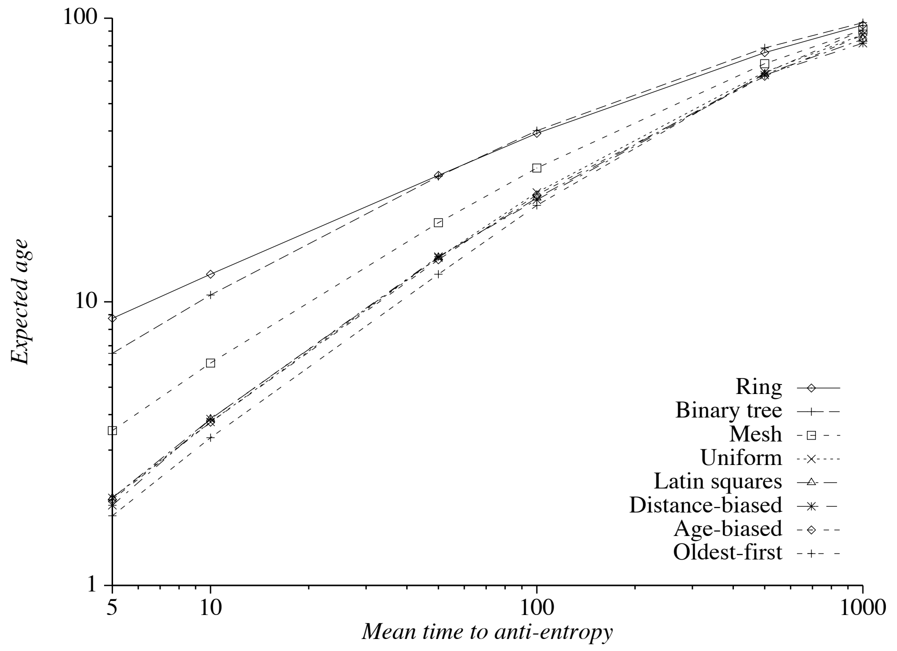

FIGURE 7: Effect of partner selection policy on expected replica age. 160 replicas; mean time-to-update 1 000. No best-effort multicast was used for this graph.

replicas cannot continue to function when disconnected from other replicas, so they are not useful for mobile computing systems.

The negative aspect of weak consistency protocols is that they allow replicas to diverge temporarily while updates are being propagated. We have found that the update propagation latency of the TSAE protocol is acceptable for many systems, and that is scales with the log of the number of replicas. Our analysis also shows that at reasonable propagation rates replicas are rarely more than a few updates behind, and that an unreliable multicast can reduce this difference further.

We are encouraged by the performance of the distance-biased partner selection policy. Similar policies can be used in the Internet to encourage traffic between nearby sites and to avoid saturating long-distance links. The random policy appears to be within a constant factor of optimal [Alon87].

Acknowledgments

John Wilkes, of the Concurrent Systems Project at Hewlett-Packard Laboratories, and Kim Taylor, of UC Santa Cruz, assisted the initial development of these protocols. George Neville-Neil, Leendert van Doorn, Bruce Montague, and Mary Long gave helpful comments on this paper.

References

[Agrawal91] D. Agrawal and A. Malpani. Efficient dissemination of information in computer networks. *Computer Journal*, **34**(6):534–41, December 1991.

[Alon87] Noga Alon, Amnon Barak, and Udi Manber. On disseminating information reliably without broadcasting. *Proceedings of the 7th International Conference on Distributed Computing Systems*, pages 74–81, 1987.

[Demers88] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. *Operating Systems Review*, **22**(1):8–32, January 1988.

[Emtage92] Alan Emtage and Peter Deutsch. Archie – an electronic directory service for the Internet. *Proceedings of the Winter Conference*, pages 93–110. Usenix Association, January 1992.

[Gifford79] D. K. Gifford. Weighted voting for replicated data. *Proceedings of the 7th ACM Symposium on Operating Systems Principles* (Pacific Grove, California), pages 150–62, December 1979.

[Golding92a] Richard Golding. End-to-end performance prediction for the Internet – progress report. Technical report UCSC–CRL–92–26. Computer and Information Sciences Board, University of California at Santa Cruz, June 1992.

[Golding92b] Richard A. Golding. *Weak-consistency group communication and membership*. PhD thesis, published as Technical report UCSC–CRL–92–52. Computer and Information Sciences Board, University of California at Santa Cruz, December 1992.

[Golding92c] Richard A. Golding and Darrell D. E. Long. Design choices for weak-consistency group communication. Technical report UCSC–CRL–92–45. Computer and Information Sciences Board, University of California at Santa Cruz, September 1992.

[Golding93] Richard A. Golding and Darrell D. E. Long. Simulation modeling of weak-consistency protocols. *Proceedings of the International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, January 1993.

[Ladin90] Rivka Ladin, Barbara Liskov, Liuba Shrira, and Sanjay Ghemawat. Lazy replication: exploiting the semantics of distributed services. Technical report MIT/LCS/TR–484. Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, July 1990.

[Lamport78] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, **21**(7):558–65, 1978.

[Lottor93] Mark Lottor. Internet Domain Survey. Technical report. Network Information Systems Center, SRI International, January 1993.

[Mockapetris87] P. Mockapetris. Domain names – concepts and facilities. Request for comments 1034. ARPA Network Working Group, November 1987.

[Oppen81] D. C. Oppen and Y. K. Dahl. The Clearinghouse: a decentralized agent for locating named objects in a distributed environment. Technical report OPD–T8103. Xerox Office Products Division, Palo Alto, Ca, 1981.

[Quarterman86] John S. Quarterman and Josiah C. Hoskins. Notable computer networks. *Communications of the ACM*, **29**(10):932–71, October 1986.

[Thomas79] R. H. Thomas. A majority consensus approach to concurrency control. *ACM Transactions on Database Systems*, **4**:180–209, 1979.

[Turek92] John Turek and Dennis Shasha. The many faces of consensus in distributed systems. *IEEE Computer*, **25**(6):8–17, June 1992.