A Simulation Evaluation of Optimistically Replicated Filing in Mobile Environments*

An-I A. Wang, Peter L. Reiher, and Rajive Bagrodia
Computer Science Department
University of California, Los Angeles

Abstract

Optimistic replication of data is becoming increasingly popular in mobile environments, but its performance and scaling characteristics are not well understood. This paper presents a simulation evaluation of optimistically replicated filing in a mobile environment. We first compare full and selective optimistic replication systems to capture the properties required for scaling. We then show that the presence of portable computers in optimistically replicated filing systems achieves a 60-percent cost reduction (e.g., computing resources) with only a 10-percent degradation of service quality (e.g., consistency of data perceived by users). This finding reveals certain similarities between the network disconnection interval and frequency of data synchronization. The research suggests new guidelines for design of optimistic replication systems.

1 Introduction

Optimistic data replication is an increasingly important technology. It allows the use of ATM banking with network failures and partitions, parallelism in making airline reservations, and simultaneous cooperative access to shared data on laptops disconnected from networks. With its resiliency to network failures, high data availability, and cooperative data sharing, optimistic replication has become one of the enabling technologies for mobile computing. Oracle 7 [2], Bayou [32], Ingres, Lotus Notes [13], Microsoft Brief Case, and Concurrent Version System are popular applications that have adopted the optimistic replication concept [19].

* This work is sponsored by the Defense Advanced Research Project Agency under contract DATB63-94-C-0080. The authors can be reached at the Computer Science Department, UCLA, Los Angeles, CA 90095, or by e-mail to {awang, reiher, rajive}@cs.ucla.edu.

Early research on optimistic replication was largely performed in the context of file systems. Most efforts were devoted to proving the correctness and practicality of the method. The Locus Operating System [23; 25], Coda [30], Ficus [7], and other projects are examples of optimistic replication. They proved that concurrent modifications can be correctly resolved without data loss [28; 18], and they deployed their systems among substantial numbers of real users [15; 21; 10]. However, the existing research sheds little light on the performance and scaling properties of optimistic replication systems, especially in the context of mobile computing.

We have constructed a validated general simulation framework [33] to evaluate the performance of optimistically replicated systems in an intermittently connected environment. This paper presents our insights on optimistic replication in terms of the properties required for scaling and behavior with the presence of portables.

Section 2 describes the motivation and mechanism of optimistic replication. Section 3 describes our simulation and input traffics. Section 4 introduces our experimental assumptions, parameter space, and metrics. Section 5 presents our insights obtained from those experiments. Section 6 relates our work to other research, and Section 7 summarizes lessons learned and makes recommendations for future optimistic replication system design.

2 Background

In large-scale distributed systems, replication is a good technique for providing high availability for data sharing across machine boundaries, because each machine can own a local copy of the data. ***Optimistic replication*** allows immediate access to any available replica of a data item, at the risk of permitting concurrent updates. In many scenarios, this risk is justifiable for three reasons. First, most files have a

single writer, particularly over a short time period. Thus, allowing writes to any data replica rarely results in concurrent updates. Second, a large class of applications (e.g., library database systems) can work well with slightly aged information, and immediate propagation of new updates to all replicas is not vital. Third, for many applications, the majority of concurrent data modifications can be performed in parallel. With proper handling, the modifications can be later merged automatically or manually without data loss. Directories are an important example. Independent file creations can be applied to two replicas of a directory without causing problems, because the differing directory replicas can be easily merged into a single directory.

Permitting copies of data to become temporarily inconsistent requires a ***reconciliation process*** to bring replicas into synchronization. This is done by comparing replicas and applying the updates at some convenient time (e.g., when portable computers are temporarily connected to the network). Typically, updates are tracked using either logging [30] or scanning [7; 27]. Improper concurrent accesses, or ***conflicts***, occur when different replicas of the same file are updated subsequent to the last reconciliation. Optimistic systems usually provide extensible application-specific libraries to resolve the majority of conflicting updates automatically. Conflicts that are not automatically resolved require user intervention [24; 28; 18].

Consider an append-only log, with log entries ordered by timestamps. If two replicas of this log receive independent updates, the optimistic replication system will allow each of them to append new records to the end of an otherwise identical log. When reconciliation between these replicas occurs, the conflict will be detected and resolved automatically by creating a log consisting of the common parts, followed by the new records in the timestamp order. Either the system or the users of the log file would provide a resolver program that could examine the conflicting versions of the file and produce the proper merged version.

On mobile computing units with limited resources, ***selective replication*** allows storage of a subset of files that will be referenced by the user in the upcoming disconnected period [26, 31]. By doing so, selective replication can lower various overheads.

3 Simulation and input traffics

Experimental studies of replication systems are few, due to various costs of obtaining useful and scalable measurements to examine a large parameter space. To overcome many of these difficulties, we have designed a general simulation framework that can be configured to evaluate large-scale replicated file systems with heterogeneous configurations. The framework comprises more than ten thousand lines of code. The built-in simulation language was developed using grammar-encoded languages, Lex and Yacc, to support compositions of simulated library components. The body of the simulation was developed using Maisie, a C-based discrete-event simulation language [1]. The simulation error is within 5 percent of the real system being calibrated, determined by the full factorial multivariate linear regression analysis [11]. Since the design and validation are applications of common software engineering disciplines and experimental methodologies, we refer the details to a prior paper [33]. However, our traffic generator is tailored to evaluate optimistic replication environments, and we will discuss this component in detail.

Our simulation input traffic is based on a three-month data trace of file access patterns, collected at Locus Computing Corporation [16]. This trace consists of the actual activities of software developers. The traced system did not use replication, but had large numbers of users and machines connected by a local area network, performing both local and remote file accesses. To a certain extent, the results presented depend on the characteristics of this workload. However, this dependency is inescapable, and we believe that the workload is realistic and representative of many office environment workloads.

Extensive pilot runs show that the approaches adopted by common traffic generators—capturing the temporal and spatial localities—are insufficient to evaluate optimistically replicated systems. In such systems, the consistency and recentness of data perceived by the end-users is highly dependent on how data are shared, as well as how data are accessed. In addition, distribution of user activities also can contribute to the overall service quality measures.

For example, our trace data reveal that as the degree of file sharing increases, the percentage of write accesses decreases. Thus, our experiments adopt a fine-grain decomposition of the file sharing and operation patterns, as shown in Figure 3.1. Without this fine-grained characterization of file sharing, the percent of write access is significantly overestimated for files at a

File access					
Read-only access	Read-write access				
	Nonshared access		Shared access		
			2-way sharing		3+way sharing
	Read access	Write access	Read access	Write access	(Further splitting)

Figure 3.1: Categorization of file accesses.

high sharing factor, and the growth rate of conflicts is potentially exponential [10].

File references fed to the model are divided into read-only accesses and read-write accesses. Read-only accesses do not affect most of the results in the simulation. Read-write file accesses are partitioned into accesses to shared and nonshared files. Assuming each user will always access a particular replica, updates to nonshared files will never create conflicts and always contain the most current data, because only a single replica is used for access to such files. Shared files are further classified as files shared between exactly two users and files shared among more than two users (e.g., two-way shared and three-plus-way shared).

To capture the daily work cycles, we model the trace at hourly intervals with mean arrival rate equal to the mean arrival rate of the corresponding hour in the trace. Average hourly traffic can be obtained by averaging the mean arrival rates of a certain hour of the day across the entire trace. Our simulation also captures the nonuniform access distribution across users by first finding the aggregate arrival rate and then distributing the aggregate traffic by the expected percentage contribution of each simulated user.

As for the patterns of disconnection, we model each portable to be connected for 2 to 12 contiguous hours during the hours with heavier file access traffics revealed by the trace. We choose long disconnected intervals instead of wireless intermittent connections because wireless network has not reached the level of ubiquity and economy. The portable is disconnected for the remainder of the day to reflect the common work schedule. For example, if we specify portables to be connected for two hours per day, each portable would randomly select two contiguous hours from the work hours, and the choice of connecting hours changes from day to day. Future work may examine the disconnection patterns with a trace directly collected from a mobile environment.

4 Experiments

4.1 Experimental assumptions

Replication granularity controls the level at which replication is applied. Possible granularities include a file, a directory, or a sub-tree of a file hierarchy (typically called a ***volume***). For simplicity, we assume that all files are replicated under a single volume, and the volume is either fully or selectively replicated across the servers. Each server contains only one replica of the volume; only one local user accesses each replica. We assume also that at most one reconciliation process is in progress at any point at a given machine. This constraint implies that if a site is participating in a reconciliation process, the site will deny reconciliation requests from other sites. All denied reconciliation requests are aborted.

Like most replication systems, each reconciliation process involves only two replicas in our model, and the site initiating replication can choose any other replica as its partner based on a specified ***reconciliation topology***. Due to space constraints, this paper only presents data on the adaptive ring topology; however, the lessons learned should be independent of topology. In a nonadaptive ring topology, all replicas participating in accessing a certain replicated item form a ring. At reconciliation time, a replica reconciles with an immediate neighboring replica in the ring. In an adaptive ring, if the target replica is busy or absent at reconciliation time, a replica will try the next immediate neighboring replica. One replica can potentially belong to many reconciliation rings because selective replication permits different replicated items in the replicated volume to be shared with different groups of replicas.

File accesses to remote replicas and various types of node and network failures are not modeled for this initial study, but will be areas of interest for future study.

4.2 Parameter space

Table 4.1 summarizes the major simulation configurations and parameters. Our simulation does not model the complexity of the operating system and underlying hardware. Those delays are modeled by parameterization of a certain measured system into analytical equations [11; 33]. In our case, we measured an early prototype of the Rumor Replicated File System (RRFS) [27, 9], running on Dell Latitude XP 486, 100 MHz laptops with 12 Mbytes of memory and a Linux

Parameters		Specifications
Environment configuration	Simulation duration	576 hours
	Calibrated hardware platform	Dell Latitude XP 486 100 Mhz (12 Mbytes of memory)
	Calibrated OS platform	Linux 2.0.x
	Network channel	10Mb Ethernet
	Physical topology	Single-level Ethernet-connected servers
System configuration	Replication system	RRFS
	File-sharing pattern	Trace-data based
	User access skewing function	Distribution mapped from the trace data
	Number of files	10150 files (from the trace; ~200 Mbytes of data)
	File size distribution	Trace-data based
	File access interval distribution	Trace-data based
	Reconciliation topology	Adaptive ring
Independent variables	Replication granularity	Full volume, selective subset of data in the volume
	Percentage of portables	0, 30, 60, 90 percent
	Connection duration for portables	2 to 12 hours per day with a 2-hour step
	Replication factor	2, 5, 8, 11, 14, 17 replicas[1]
	Reconciliation interval	1 to 21 hours with a 3-hour step[1]

Table 4.1: Major simulation parameters.

Metrics		Definitions
Cost metrics	Reconciliation time	Elapsed time from the moment a replica requests reconciliation with a remote replica to the moment that all files in the local replica are reconciled, excluding aborted sessions[2]
	CPU overhead	Percent CPU utilization per replica due to reconciliation processes, including the aborted sessions
	Transmission volume per replica	Bandwidth required per replica for periodic reconciliation processes
	Aggregate transmission volume	Global bandwidth required for all replicas to execute periodic reconciliation processes
	Storage	Storage required per replica to keep track of the replicated states
QoS metrics	Conflict rate	Number of conflicts per replica per hour
	Time to stale read	Expected number of days before encountering one stale read
	Time to stale write	Expected number of days before encountering one stale write

Table 4.2: Definitions of various metrics.

1.2.x operating system. Certain results (e.g., running time of the system) are thus contingent on the versions of the software and hardware we chose. The input traffic patterns are extracted from the Locus trace with the methods we described in Section 3.

For the independent variables, we conducted more than 2000 pilot runs over a large parameter space with various resolutions to locate the variables and regions of high variability for further investigations. The parameter space and resolution were reduced when necessary to accelerate the data collection process.

Our simulation investigated the coarsest possible granularity—full volume, and the finest possible granularity—selective replication, with perfect prediction of upcoming file references. To examine the effects of portable users, we varied the percentage of portable users from 0 to 90 percent. The connection duration for those portables was varied from 2 to 12 hours according to the description in Section 3. ***Replication factor***, or the number of data copies, was varied from 2 to 17 replicas.

Reconciliation interval describes the frequency of the reconciliation process. In most systems, reconciliation is performed immediately after an update [8], periodically [27], or on demand [30]. The results presented in this paper use periodic reconciliation only, with no fast propagation at update time. We varied the interval from 1 to 21 hours.

4.3 Metrics

Table 4.2 defines various cost and quality of service (***QoS***) metrics. The cost metrics are fairly straightforward. For QoS, one popular metric is the conflict rate. However, as shown in [33, 17], this metric is flawed. Consider the scenario where detection and

[1] Seventeen replicas would take approximately one hour to complete a full volume reconciliation, which imposes the minimum reconciliation interval to be one hour.

[2] The time spent for aborted sessions is the elapsed time from the moment at which a replica requests reconciliation with a remote replica to the moment at which the replica receives the rejection message from the remote replica. This metric excludes the aborted sessions because it measures the average execution time for a successful reconciliation process.

recording of conflicts occur only during a reconciliation process. We can reduce the number of conflicts to zero by never reconciling, resulting in no update ever being propagated anywhere. Despite its popularity, we will omit the conflict-rate metric in this paper. Instead, we use ***time to stale read*** and ***time to stale write*** metrics. A stale read or stale write occurs whenever a user reads or writes an out-of-date replica of a file. These metrics are hard to measure in a real system, as they depend on global knowledge, but they can easily be measured in our simulation.

5 Results

The scaling of optimistic replication systems is essential for their widespread use. Section 5.1 first explores the properties required for scaling such mobile systems, and Section 5.2 presents the effects of using portable computers in the context of selective replication.

5.1 Scaling properties of optimistic replication

We evaluate scaling from the perspective of both implementation and real usage patterns. Figure 5.1 compares the running time required to perform reconciliation for full and selective replication methods. The full replication shows linear scaling, and the selective replication shows near-constant scaling beyond five replicas.

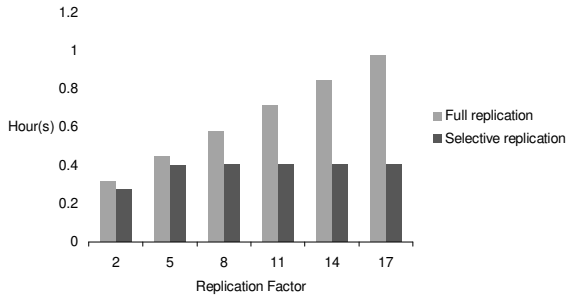

Figure 5.1: Reconciliation time for full optimistic replication and selective optimistic replication. Standard deviation (not clearly visible) is within 1 percent for each data point.

Selective replication can achieve near constant scaling because it exploits the user pattern, which shows that the majority of files are not shared beyond four replicas, and selective replication stores the state information only for replicas participating in shared data accesses. On the other hand, full replication stores the state information for all replicas, and the processing overhead is directly proportional to the replication factor.

Intuitively, high-precision and fine-grained targeting of selective replication should yield minimum costs and the highest QoS. Table 5.1 shows differently. Selective replication is indeed cheaper. The reconciliation time is very likely to stay within half an hour for 50 to even 100 replicas. For CPU utilization, selective replication saves 68 percent; for transmission volume, 16 percent or 2.8 minutes on 14.4-Kbyte modems; for aggregate network bandwidth, 18 percent; and for storage, 47 percent. However, selective replication loses big on the QoS. For times to stale read and stale write, selective replication loses 45 and 47 percent correspondingly.

Metrics	Costs[3]	
	Full rep.	Sel. rep.
Recon. time (min.)	38	24
CPU overhead (during the recon. time)	19%	6.0%
Trans. vol. per replica (Mbytes/hr)	1.9	1.6
Aggregate trans. vol. (Mbytes/hr)	17	14
Storage per replica (Mbytes/rep)	19	10
Time to stale read (days/stale read)	12	6.3
Time to stale write (days/stale write)	22	12

Table 5.1: Comparison of full optimistic replication and selective optimistic replication under various predefined metrics.

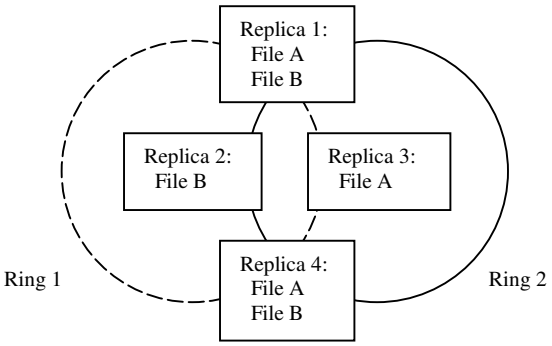

Figure 5.2: Example of better QoS by full replication than selective replication.

[3] Median values are presented because of the skewing of various data curves. For each replication method, the median values resemble the simulation environment of 30 to 60 percent portable users with 8 hours of daily connection duration, 8 to 11 replicas, and 9- to 13-hour reconciliation intervals.

Metrics	Percentages of portable computers[4]			
	0%	30%	60%	90%
Reconciliation time (minutes)	24	24	24	24
CPU overhead (during the reconciliation time)	8.2%	6.8%	3.9%	2.9%
Transmission volume per replica (Mbytes/hr)	2.3	1.9	1.0	0.89
Aggregate transmission volume (Mbytes/hr)	22	19	11	8.0
Storage per replica (Mbytes/rep)	10	10	10	10
Time to stale read (days/stale read)	6.8	6.4	5.8	5.9
Time to stale write (days/stale write)	12	12	12	11

Table 5.2: Effects of various percentages of portable computers on predefined metrics.

Metrics	Hours of connection per day[5]							
	0	2	4	6	8	10	12	24
Reconciliation time (minutes)	0	24	24	24	24	24	24	24
CPU overhead (during the reconciliation time)	0.0%	3.3%	4.2%	5.6%	6.0%	6.3%	6.3%	8.2%
Transmission volume per replica (Mbytes/hr)	0.0	0.92	1.3	1.5	1.6	1.8	1.9	2.3
Aggregate transmission volume (Mbytes/hr)	0.0	6.5	8.6	13	14	16	17	22
Storage per replica (Mbytes/rep)	0	10	10	10	10	10	10	10
Time to stale read (days/stale read)	3.5	4.9	5.4	6.0	6.3	6.4	6.4	6.8
Time to stale write (days/stale write)	9.3	10	11	11	12	12	12	12

Table 5.3: Sensitivity of laptop network connection period on the predefined metrics.

With some investigation, we found that the extra work performed by full optimistic replication actually assists the indirect propagation of data not shared by the local replica. Figure 5.2 illustrates a scenario using the adaptive ring

Adaptive ring 1 (replica 1, 3, and 4) shares file A, and adaptive ring 2 (replica 1, 2, and 4) shares file B. By the definition of selective optimistic replication, when replica 1 and 2 reconcile, only file B is synchronized; similarly, when replica 1 and 3 reconcile, only file A is synchronized. For both file A and file B to reach replica 4, assuming no failures and no other data propagation paths are available, selective replication requires a minimum of four reconciliation processes, but full replication requires only two. Full replication can perform better when the sharing groups are densely interconnected. Our QoS result suggests dense interconnection of sharing groups in the trace because the QoS greatly benefits from the redundant data paths.

The number of states stored by the system and the actual degree of sharing by the users have a strong influence over replication costs; the degree of freedom for update dissemination and the amount of overlapping among the sharing group affect the replication QoS. Combining the two is essential for large-scale and high-quality optimistic replication service.

5.2 Desirable system characteristics for portable computers

To study mobile computing, we have simulated a scenario with various percentages of portable computers, with a mean connection duration of eight hours to reflect a common daily work cycle. Table 5.2 shows the effects of various percentages of portable computers on our predefined metrics.

These numbers are encouraging; the presence of disconnected units alleviates the replication workload. The reconciliation time and storage remain constant. The CPU utilization rate can drop by up to 65 percent; transmission volume, 61 percent (a 13-minute reduction on 14.4-Kbyte modems); and aggregate network bandwidth, 64 percent. These reductions are all due to fewer reconciliations being performed, since disconnected machines cannot reconcile. On the other hand, the QoS loss is relatively small, and the degradation is graceful. Times to stale read and stale write degrade by 13 and 8.3 percent, respectively.

We have also investigated the effects of connection duration from 2 to 12 hours. Table 5.3 shows the

[4] Median values are presented because of the skewing of various data curves. The median values resemble the simulation environment of 8 to 11 replicas and 9- to 11-hour reconciliation intervals.

[5] Median values are presented because of the skewing of various data curves. The median values resemble the simulation environment of 30 to 60 percent portables, 8 to 11 replicas, and 9- to 13-hour reconciliation intervals.

effects of varying connection duration on our predefined metrics. The 0-hour case represents the worse QoS, where no reconciliation occurs for *any* replicas, and the 24-hour case represents the best case with the given parameter settings.

The QoS numbers in the worse case appear to be high and suggest that the shared accesses are relatively infrequent or highly localized. We believe that the locality is the primary cause, because 13 percent of trace accesses are shared accesses. The QoS numbers in the best case set the upper bound for the QoS we can achieve with daylong connectivity; times to stale read and stale write can improve by 94 and 29 percent respectively. Various cost and QoS metrics show asymptotic behavior as the connection duration lengthens. Notably, QoS improves little beyond eight hours of connection duration. This result complies with our intuition that longer connection duration beyond working hours should not contribute significantly to QoS.

From a different angle, network disconnection duration does introduce complexities into optimistic replication systems. If the reconciliation interval is one hour and the disconnection period is one week, or if the reconciliation interval is one week and the disconnection period is one hour, the net effect for both cases is equivalent to using a reconciliation interval of one week. Whenever both reconciliation interval and disconnection coexist, the longer duration of the two dominates. In our experiments, effectively we have two sets of computers (portables and non-portables) reconciling at two different rates. Future investigation requires repartitioning of the problem space, to find simpler relationships among the reconciliation interval, disconnection patterns, and other metrics.

6 Related work

Many analytical modeling and simulation studies have been done to compare the performance of various replication control protocols, but optimistic consistency strategies are often not addressed [5; 12; 20].

Golding's thesis [4] investigates optimistic replication in the static scenario without considering disconnected operations. Golding describes a timestamped anti-entropy protocol that has a one-to-one mapping to our reconciliation process. His primary QoS metric is the mean time to convergence of concurrent updates [3]. One major concern with his approach is the lack of updates during the convergence period, and his results do not capture the effects of user patterns. This static assumption precludes the dynamic possibility that the update rate of users can exceed the convergence rate. Under such a scenario, the system can reach a highly unstable state.

Scaling of optimistic replication is a highly controversial topic. Gray and associates [6] have expressed some of the strongest warnings against optimistic replication. Through the analytical approach and assumption of uniform access patterns of replicated data, they suggest that the scaling of optimistic replication is questionable. However, from our prior study [33], we have shown that the temporal access locality alone can improve the QoS by an order of magnitude. From the perspective of data sharing, 81 percent of our files are only two-way shared among the users, and the overall system scales with minor degradation of service quality. Similar to this uniformity assumption, Gray and associates made other assumptions on the uniform usage patterns, which depict unrealistic and overly pessimistic predictions.

Kistler and Satyanarayanan [15] at CMU have conducted an empirical study of disconnected operations of Coda, which utilizes the limited optimistic replication capabilities in the Andrew File System [14] to offer optimistic replication and caching support for operation with network disconnection. The primary interest of their study is to prove the feasibility of disconnected operation. With trace data, researchers demonstrated the feasibility by showing that the average working set is small and adequate for caching. They also demonstrated the low likelihood of concurrent updates [15; 22]. However, simulation is still the primary venue to explore large parameter space and hypothetical scenarios. We are also seeking opportunities to analyze existing trace data from various sources to formulate the network disconnection patterns in our future work.

7 Lessons learned and recommendations

We have compared full optimistic replication and selective optimistic replication schemes to characterize requirements for scaling. Future optimistic replication systems should exploit the low degree of sharing present in user patterns to reduce system costs, and the interconnection among the replicas should be sufficiently dense to leverage the indirect data propagation. However, for the selective replication case, we also see that aggressive local cost optimization can actually reduce the opportunity for indirect update propagation and degrade global QoS. Future system design should consider both local and global system behaviors. For example, one candidate for our future study is to consider selective optimistic replication that stores the states for the top 10 percent of frequently

modified files. This 10 percent increase of overhead may accomplish 90 percent of the benefits of full optimistic replication.

Another finding is that the presence of portable computers alleviates the resource costs by 60 percent, and the degradation of QoS is only 10 percent. This finding also reveals the similarities between network disconnection and reconciliation interval, and suggests further investigation.

Finally, from full and selective replication experiments we have found that no single configuration can win in both areas of cost and QoS. However, it is our first step toward understanding and designing a cost-effective, high-QoS optimistic replication system for mobile computing environments.

8 References

[1] Bagrodia R, Laio WT. MAISIE: A Language for the Design of Efficient Discrete-Event Simulations. *IEEE Transactions on Software Engineering*, 20(4): 225-238, April 1994.

[2] Daniels D, Doo LB, Downing A, Elsbernd C, Hallmark G, Jain S, Jenkins B, Lim P, Smith G, Souder B, Stamos J. Oracle's Symmetric Replication Technology and Implications for Application Design. *Proceedings of SIGMOD Conference*, p. 467, 1994.

[3] Golding RA, Long DDE. Accessing Replicated Data in a Large-Scale Distributed System. International Journal in Computer Simulation, 1(2), 1991.

[4] Golding RA. Modeling Replica Divergence in a Weak-Consistency Protocol for Global Scale Distribution Data Bases. Technical report UCSC-CRL-93-09, University of California, Santa Cruz, 1993.

[5] Goldweber M, Johnson DB, Raab L. A Comparison of Consistency Control Protocols. Technical report PCS-TR89-142, Department of Mathematics and Computer Science, Dartmouth College, 1989.

[6] Gray J, Helland P, O'Neil P, Shasha D. The Dangers of Replication and a Solution. *Proceedings of the 1996 ACM SIGMOD Conference*, pp.173-182, 1996.

[7] Guy R, Heidemann J, Mak W, Page T, Popek G, Rothmeier D. Implementation of the Ficus Replicated File System. *Proceedings of the Usenix Summer Conference*, pp. 63-71, June 1990.

[8] Guy R, Popek G, Page TW. Consistency Algorithms for Optimistic Replication. *Proceedings of the First International Conference on Network Protocols*, *IEEE*, October 1993.

[9] Guy R, Reiher P, Ratner D, Gunter M, Ma W, Popek G. Rumor: Mobile Data Access Through Optimistic Peer-to-Peer Replication. *Proceedings of Workshop on Mobile Data Access*, November 1998.

[10] Heidemann J, Goel A, Popek G. Defining and Measuring Conflicts in Optimistic Replication. Technical report CSD-950033, University of California, Los Angeles, 1995.

[11] Jain R. *The Art of Computer Systems Performance Analysis*. New York, John Wiley's, 1991.

[12] Johnson DB, Raab L. A Tight Upper Bound on the Benefits of Replication and Consistency Control Protocols. Technical report PCS-TR90-157, Department of Mathematics and Computer Science, Dartmouth College, 1990.

[13] Kawell LJ, Beckhardt S, Halvorsen T, Ozzie R, Greif I. Replicated Document Management in a Group Communication System. Groupware*: Software for Computer-Supported Cooperative Work*, IEEE Computer Society Press, 1992, pp. 226-235.

[14] Kazar M. Synchronization and Caching Issues in the Andrew File System. *Proceeding of the Winter Usenix Conference*, pp. 31-43, February 1998.

[15] Kistler JJ, Satyanarayanan M. Disconnected Operation in the Coda File System. *ACM Transactions on Computer Systems*, 10(1), February 1992.

[16] Kuenning GH, Popek GJ, Reiher PL. An Analysis of Trace Data for Predictive File Caching in Mobile Computing. *Proceedings of the 1994 Summer Usenix Conference, 1994.*

[17] Kuenning GH, Bagrodia R, Guy RG, Popek GJ, Reiher P, Wang A. Measuring the Quality of Service of Optimistic Replication. *Proceedings of the ECOOP Workshop on Mobility and Replication*, July, 1998.

[18] Kumar P, Satyanarayanan M. Flexible and Safe Resolution of File Conflicts. *Proceedings of the 1995 Usenix Technical Conference*, pp. 95-106, January 1995.

[19] Kung HT, Robinson J. On Optmistc Methods for Concurrency Control. *ACM Transactions on Database Systems*, 6(2), June 1981.

[20] Liu ML, Agrawal D, Abbadi AE. What Price Replication? Technical report TRCS94-14, Computer Science Department, University of California, Santa Barbara, July 1994.

[21] Noble BD, Satyanarayanan M. An Empirical Study of a Highly Available File System. *Proceedings of the 1994 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, Nashville, TN, May 1994.

[22] Page T, Guy R, Heidemann J, Ratner D, Reiher P, Goel A, Kuenning G, Popek G. Perspectives on Optimistically Replicated, Peer-to-Peer Filing. *Software—Practice and Experience*, December 1997.

[23] Parker DS, Popek G, Rudison G, Stoughton A, Walker B, Walton E, Chow J, Edwards D, Kiser S, Kline C. Detection of Mutual Inconsistency in Distributed

Systems. *IEEE Transactions on Software Engineering*, pp. 240-247, May 1983.

[24] Popek GJ, Guy RG, Page TW, Heidemann JS. Replication in Ficus Distributed File Systems. *Proceedings of the Workshop on Management of Replicated Data*, Houston, Texas, November 1990.

[25] Popek GJ, Walker B. *The Locus Distributed Operating System*, MIT Press, 1985.

[26] Ratner D. Selective Replication: Fine-Grain Control of Replicated Files. Masters Thesis. University of California, Los Angeles, 1995.

[27] Ratner D, Popek GJ, Reiher P. The Ward Model: A Scalable Replication Architecture for Mobility. *Proceedings of the OOPSLA'96 Workshop on Object Replication and Mobile Computing (ORMC'96)*, San Jose, California, October 7, 1996.

[28] Reiher P, Heidemann J, Ratner D, Skinner G, Popek G. Resolving File Conflicts in the Ficus File System. *Proceedings of USENIX Conference*, pp. 183-195, June 1994.

[29] Reiher P, Popek J, Gunter M, Salomone J, Ratner D. Peer-to-Peer Reconciliation-Based Replication for Mobile Computers. *ECCOP 1996 Second* Workshop on Mobility and Replication, July 1996.

[30] Satyanarayanan M. Coda: A Highly Available File System for a Disconnected Workstation Environment. *Proceedings of the Second Workshop on Workstation Operating Systems*, September 1989.

[31] Sweeden B. *Lotus Notes 4.5 Administrator's Guide*, Sybex Network Press, 1997.

[32] Terry DB, Theimer MM, Petersen K, Demers AJ, Spreitzer MJ, Hauser CH. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. *Proceedings of the 15th ACM Symposium on Operating Systems Principle*, December 1995.

[33] Wang AI. A Simulation Evaluation for Optimistically Replicated Environment. Master's Thesis. University of California, Los Angeles, 1998.