A Realistic Evaluation of Consistency Algorithms for Replicated Files

Jehan-François Pâris
Darrell D.E. Long
Alexander Glockner

Computer Systems Research Group
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093

Abstract: Data are often replicated in distributed systems to protect them against site failures and network malfunctions. When this is the case, an access policy must be chosen to insure that a consistent view of the data is always presented. Voting protocols guarantee consistency of replicated data in the presence of any scenario involving non-Byzantine site failures and network partitions. While *Static Majority Consensus Voting* protocols use static quorums, *Dynamic Voting* protocols, like *Dynamic Voting* and *Lexicographic Dynamic Voting,* dynamically adjust quorums to changes in the status of the network of sites holding the copies.

The availabilities of replicated data managed by these three protocols are compared using a simulation model with realistic parameters. Dynamic Voting is found to perform better than Majority Consensus Voting for all files having more than three copies while Lexicographic Dynamic Voting performs much better than the two other protocols for all eleven configurations under study.

Keywords: discrete-event simulation, file consistency, fault-tolerant systems, replicated files, batch-means.

1. INTRODUCTION

In a distributed system, the data are often replicated for protection against site failures and network partitions. By maintaining several copies of the same file on different sites, increased availability and reliability of access can be obtained. Recent technological improvements in computer networks and reductions in the cost of storage media have made the replication of important files a viable proposition.

When files are replicated, an access policy must be chosen to insure that a consistent view of the data is always presented. Several methods have been discussed in the literature, including token based schemes, active copy schemes and schemes based on quorum consensus. This paper focuses on protocols that are based on quorum consensus because of their simplicity and their tolerance of network partitions.

Static consensus protocols, like Majority Consensus Voting [Giff79], do not provide the highest availability possible.

Dynamic Voting [DaBu85] improves upon majority consensus voting by allowing quorums to be adjusted automatically during system operation. An extension [Jajo87] to Dynamic Voting improves file availability by imposing a total ordering on the sites in order to break ties. Until now there has been no comparative study of these three protocols.

†This work was sponsored in part by grants from U.C. MICRO Program and NCR Corporation.

Stochastic process models have been widely used to evaluate consistency protocols. Unfortunately, many difficulties prevent relying solely upon them. Stochastic models quickly become extremely complex when general (non-exponential) time distributions are considered; site repair times are better represented by a general distribution. The problem of modeling network partitions and site failures simultaneously is intractable for all but the most basic cases [NKT87]. Algebraic expressions for file reliability are difficult to obtain, even for the simplest algorithms and site configurations. For these and other reasons simulation has been chosen as a method to evaluate the performance of the consistency protocols.

The remainder of this paper is organized as follows: in Section 2, the three quorum consensus protocols to be evaluated are reviewed; in Section 3, the network model is described; and in Sections 4 and 5, results and conclusions are presented.

2. SURVEY OF CONSISTENCY ALGORITHMS

In its simplest form, *Majority Consensus Voting* [Elli77, Giff79, ElFl83] assumes that the current state of a replicated file is the state of the majority of its copies. Different quorums can be defined and different weights, including none, allocated to each copy. Consistency is guaranteed so long as the quorum is high enough to forbid access to two disjoint subsets of the copies.

Majority Consensus Voting has the disadvantage that it only allows access to a replicated file when a majority of its copies are available. *Dynamic Voting* [DaBu85, BGS86] and *Lexicographic Dynamic Voting* [Jajo87] overcome this limitation by adjusting the quorum to reflect changes in the number of accessible physical copies. It has been shown [PaBu86] that dynamic voting protocols significantly improve the availability of replicated files over static consensus protocols.

Dynamic Voting protocols guarantee file consistency as long as all sites not operating correctly immediately cease operation. Sequenced, reliable message delivery is assumed. To keep track of the status of the replicated file, every physical copy of a replicated file will maintain some state information. This information will include a *version number* and a *partition vector.* The version number x_i of a physical copy on site i is a positive integer that identifies the last successful update recorded by that copy. The partition vector z^i of a physical copy on site i is a vector whose elements $z^i{}_j$ are equal to zero if sites i and j can communicate and have otherwise the value x_i of the version number of the physical copy on i at the time i and j became unable to communicate. All $z^i{}_i$ will always be zero. Partition vectors will be automatically updated on every site at the time of every network failure. They will be used by the dynamic voting protocol to check if a set of current physical copies that can communicate amongst themselves are a majority of the previous quorum of copies. For this reason, that set is referred to as a *majority block.* Once such a set is established, writing to any disjoint set of copies is forbidden, so the set can become the new quorum.

To illustrate these concepts, consider a replicated file consisting of three physical copies located at sites A, B and C. Assuming that all sites and all links are operational, the initial version numbers x_i are 1 and the partition vectors z^i are $(0,0,0)$ for all three copies:

A	B	C
$x_A = 1$	$x_B = 1$	$x_C = 1$
$z^A = (0,0,0)$	$z^B = (0,0,0)$	$z^C = (0,0,0)$

The initial majority block consists of all three copies *A*, *B* and *C*. After five update operations are successfully completed, the state of the replicated file is represented by:

A	B	C
$x_A = 6$	$x_B = 6$	$x_C = 6$
$z^A = (0,0,0)$	$z^B = (0,0,0)$	$z^C = (0,0,0)$

The majority block still consists of all three copies *A*, *B* and *C*. Suppose now that site *B* fails. The state of the replicated file is now:

A	B	C
$x_A = 6$	$x_B = 6$	$x_C = 6$
$z^A = (0,6,0)$	$z^B = (0,0,0)$	$z^C = (0,6,0)$

The block consisting of sites *A* and *C* contains a majority of the sites included in the previous majority block, and *A* and *C* can communicate. That block will therefore become the new majority block.

Assume that the link between *A* and *C* fails after three additional update operations have taken place. The network is now partitioned into the two disjoint subsets {*A*} and {*C*}. The file will then be in the following configuration:

A	B	C
$x_A = 9$	$x_B = 6$	$x_C = 9$
$z^A = (0,6,9)$	$z^B = (0,0,0)$	$z^C = (9,6,0)$

There is exactly one site of the previous majority block on each side of the partition. When such ties occur, dynamic voting cannot proceed and declares that the replicated file is *not* available as long as a new majority block including a majority of sites from the last majority block cannot be created. To return to the example, *A* and *C* need to be able to communicate again before the file becomes available.

Such situations, where the number of current physical copies within a group of mutually communicating sites is equal to the number of current copies not in communication, are not infrequent. Lexicographic Dynamic Voting [Jajo87] accommodates these situations by introducing a tie breaking rule. The sites are given a static linear ordering. Then, when the largest group of communicating sites contains exactly one-half of the current copies in the last majority block, it will be declared the new majority block if and only if it contains the "largest" site in the previous majority block.

In the previous example, suppose the sites had been ordered so $A < B < C$. Then, after the link between *A* and *C* failed, site *C* by itself could be the new majority block. Site *C* can determine this itself by consulting its partition vector for the file. It can determine that it cannot communicate with site *A* and that the previous majority block consists of the subset {*A* , *C* }. Since *C* ranks higher than *A*, the group containing C is the majority block. By the same reasoning, site A determines that it is not the majority block.

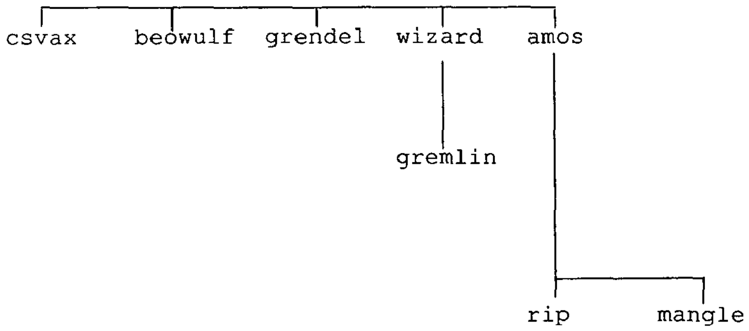

Figure 1: Network Topology

3. THE SIMULATION MODEL

The goal of this study was to model as accurately as possible the behavior of consistency protocols in a file replication scheme involving several sites on a local area network. The model consists of eight sites and three linked CSMA/CD subnets from the UCSD Computer Science Department to capture the realism of such an environment. Each of the sites is able to store and manipulate copies of a file.

Site failure and repair data are summarized in Table 1. Individual values for mean time to failure, percentage of hardware faults, repair times for hardware and software failures, and preventive maintenance schedules were chosen to reflect as accurately as possible the true behavior of the sites modeled. Exponential failure distributions were chosen for all eight sites. Hardware failures normally need human intervention and often require a service call. Hardware repair times were modeled by a constant term representing the minimum service time plus an exponentially distributed term representing the actual repair process. Since software failures only require a system restart, constant recovery times are assumed.

The three subnets are assumed not to fail; gateway sites may fail resulting in network partitions. Message delivery is guaranteed to all active sites in the current partition when a file access request is made. Local experience justifies these assumptions.

Five of the sites are connected on the main subnet. One of these sites is the gateway to the second subnet, to which the sixth site is connected; another of the five sites is the gateway to the third subnet, to which the seventh and eighth sites are connected.

Access to the replicated file is modeled as a single user that can access any of the eight sites. The access requests are granted or refused based solely on the current state of the sites containing copies and the capability of the protocol to guarantee file consistency.

The model was programmed in SIMSCRIPT II.5 and run on a VAX 11/780. The process interaction approach [Fish78] was chosen, since the sites and the user are easily described as independent processes.

Table 1: Site Characteristics

Site	Name	Mean Time To Fail (days)	Hardware Failures (%)	Restart Time (min.)	Hardware Repair Time	
					Constant Part (hours)	Exp. Part (hours)
1	csvax	36.5	10	20.0	0	2
2	beowulf	10	10	15	4	24
3	grendel	365	90	10	0	2
4	wizard	50	50	15	168	168
5	amos	365	90	10	0	2
6	gremlin	50	50	15	168	168
7	rip	50	50	15	168	168
8	mangle	50	50	15	168	168

Note: Sites 1, 3 and 5 are unavailable for 3 hours every 90 days for preventive maintenance.

All simulations were started with all sites operating and a time-to-steady-state interval of 360 days; these were considered acceptable by noting the time spent in each state and by examining graphs of the measured values in the early history of the simulations. The simulations were run for 400 simulated years, long enough to establish a 95% confidence interval for all file unavailabilities with an interval size listed in Table 2.

To analyze the output data, batch-means analysis was chosen for its simplicity and its general applicability [LaMi87]. After the initial bias was removed by ignoring the statistics gathered during the time-to-steady-state interval, a very long simulation run was divided into time segments of equal size called *batches*. Pairs of batches were then merged until the resulting batches could be considered statistically independent of each other. The final batch size was chosen using a heuristic developed by Law and Carson [LaCa79]; after finding a batch size *m* where the batches have a correlation of 0.4 or less, the batches of size 10*m* could be considered uncorrelated.

4. DISCUSSION OF THE RESULTS

Eleven configurations were considered in the study. The first four consist of three copies. Configuration A is comprised of copies on sites 1, 2 and 4, which allows for no partitions. Configuration B consists of copies on sites 1, 2 and 6 with a single partition point at site 4. Configuration C has copies on sites 1, 2 and 8 with one partition point at site 4 and another at site 5. Configuration D is comprised of copies on sites 6, 7 and 8; either site 4 or 5 can cause a partition. Four other configurations consist of four copies distributed as follows. Configuration E has copies on sites 1, 2, 3 and 4, which allows for no partitions. Configuration F is comprised of copies on sites 1, 2, 4 and 6 with a partition point at site 4. Configuration G has copies on sites 1, 2, 6 and 8 with partition points at sites 4 and 5. Configuration H consists of two pairs of copies at sites 1 and 2 and sites 7 and 8 separated by a single partition point at site 5. The last three configurations have five copies. Configuration I is comprised of copies on sites 1, 2, 3, 4 and 5, which allows for no partition. Configuration J consists of copies on sites 1, 2, 3, 7 and 8 separated by a single partition point at site 5. Finally, configuration

Table 2: Confidence Intervals for File Unavailability

unavailability	interval width
< 0.0001	± 0.00002
< 0.001	± 0.0004
< 0.01	± 0.0002
< 0.1	± 0.005
< 1.0	± 0.01

K has copies on sites 1, 2, 6, 7 and 8 with partition points at sites 4 and 5.

Configurations with six or more sites were not considered as it was assumed that the update traffic costs associated with such configurations would make them highly unlikely in the kind of environment under investigation.

In Table 3, the unavailabilities of replicated files for all eleven configurations and all three consistency protocols are summarized. It was decided to measure and display unavailabilities since they indicate more clearly the differences among the protocols.

The first finding was that Dynamic Voting (DV) performed worse than Majority Consensus Voting (MCV) for three copies. This is not surprising since the same conclusion had already been reached by Pâris and Burkhard using Markov chains [PaBu86]. Dynamic Voting requires at least two copies from the previous majority block to form a new majority block and is more restrictive than Majority Consensus Voting which only requires only two copies in this case. The same is not true of Lexicographic Dynamic Voting (LDV), which clearly outperforms MCV for three copies.

For four copies, it was found that Dynamic Voting performed much better than Majority Consensus Voting in configurations E and G where partitions are either not allowed or not likely to cause ties. The situation was different for configurations F and G where the failure of a single site could result in a tie. For instance, the failure of site 5 in configuration G will normally leave the system with two operational groups of the same size. The unavailability of the configuration is not essentially different from the unavailability of a replicated file consisting of a single copy at site 5. Lexicographic Dynamic Voting, which resolves ties, outperforms Majority Consensus Voting and Dynamic Voting for all four configurations with four copies.

Similar conclusions were also found for five copies. The only real surprise was the bad performance of Majority Consensus Voting for configuration K. Comparing the unavailabilities obtained by Majority Consensus Voting in configurations H and K, it was found that this protocol performs worse when site 6 is added to sites 1, 2, 7 and 8. The reason for this apparent paradox is simple: configuration K has three relatively unreliable sites, 6, 7 and 8, and two partition points, 4 and 5, the first of which is also relatively unreliable. Since three copies are now needed to obtain a quorum, the failure of sites 7, 8, and 4 or 6 is now sufficient to make the file unavailable. A lower bound for the unavailability of configuration K under Majority Consensus Voting is then given by

$$U_7U_8(U_4+U_6) = 0.003595$$

where U_i designates the unavailability of site i.

The mean length of time that a replicated file was unavailable for every protocol and every configuration were also measured. These figures are summarized in Table 4.

Table 3: Replicated File Unavailabilities

Sites	Consistency Policy		
	MCV	DV	LDV
A: 1, 2, 4	0.002130	0.004348	0.000668
B: 1, 2, 6	0.003871	0.008281	0.001214
C: 1, 6, 8	0.031127	0.056428	0.001707
D: 6, 7, 8	0.069342	0.117683	0.053592
E: 1, 2, 3, 4	0.000608	0.000018	0.000010
F: 1, 2, 4, 6	0.002761	0.108034	0.002154
G: 1, 2, 6, 8	0.002027	0.001510	0.000151
H: 1, 2, 7, 8	0.001408	0.004275	0.000365
I: 1, 2, 3, 4, 5	0.000025	0.000000	0.000000
J: 1, 2, 3, 7, 8	0.000388	0.000057	0.000000
K: 1, 2, 6, 7, 8	0.005310	0.001094	0.000018

Table 4: Mean Duration of Unavailable Periods

Sites	Consistency Policy		
	MCV	DV	LDV
A: 1, 2, 4	0.101968	0.210651	0.077353
B: 1, 2, 6	0.101059	0.217369	0.078867
C: 1, 6, 8	0.944336	1.868895	0.085960
D: 6, 7, 8	3.000469	5.850864	7.850864
E: 1, 2, 3, 4	0.071134	0.063630	0.062950
F: 1, 2, 4, 6	0.102001	5.962853	0.275006
G: 1, 2, 6, 8	0.084714	0.297879	0.077870
H: 1, 2, 7, 8	0.078933	0.142206	0.245300
I: 1, 2, 3, 4, 5	0.06231	-	-
J: 1, 2, 3, 7, 8	0.079213	0.118580	-
K: 1, 2, 6, 7, 8	0.260999	0.149210	0.06880

5. CONCLUSION

This paper presented a simulation model aimed at comparing the performance of several consistency protocols for replicated files in real-life situations. The model was based on an extant system allowing for network partitions. Failure and recovery parameters for individual sites were selected to reflect several years of experience with this network. Unlike previous studies, non-exponential repair times were considered as it was found that they modeled more accurately the life cycles of actual machines.

Since network partitions were taken into account, cases were found where ties degraded the performance of Dynamic Voting below that of Majority Consensus Voting. Lexicographic Dynamic Voting was found to outperform its two competitors in all situations. All these results confirm and extend the results obtained by Burkhard, Long and Pâris using Markov chains as described in [BuPa86] and [LoPa87]. They confirm that Lexicographic Dynamic Voting is a superior protocol and constitute a prime motivation for research into efficient implementations.

Many tasks remain to be completed. One of them is a better definition of file availability. As in previous papers [LoPa87, CLP87], we assumed that a replicated file was available as long as it could be accessed from at least one location on the net. This definition does not take into account the fact that a network partition might have left some users without access to the sites in the current majority; the replicated file is effectively unavailable for them although it is still available for others. A possible alternative would be to define file availability as the the fraction of attempted file accesses that are successfully completed. The measured availability of a replicated file would then be an estimate of the perceived availability of the file and would be dependent on user locations and file access patterns.

Availability measurements should also have been concerned with the transient behavior of the systems instead of their steady-state performance. Distributed file systems are essentially evolving entities. The data we gathered concerned the behavior of file systems operating without disturbances for very long periods of time.

A more practical problem is the lack of reliable data on machine failures and repairs. Failure rates and repair time distributions used in this study were estimates instead of direct measurements as they should have been. To address this issue, we are currently developing distributed tools to gather data about site individual failures and recoveries in a local area network of UNIX machines.

Finally, our model did not address the issue of the overhead incurred by consistency protocols in terms of message traffic overhead and time delays. It is not clear however that simulation is the best tool to gather such data. Message traffic overheads are easy to estimate by standard probabilistic methods [CLP87]. Time delays can be more accurately assessed by direct measurements on a properly instrumented testbed, like the Gemini replicated file testbed system [BMP87].

Within these limitations, discrete event simulation nevertheless constitute the tool of choice for analyzing the behavior of replicated data objects in distributed environments. Unlike stochastic models, it sets no restrictions on the sizes, topologies and repair modes of the systems under study. Unlike direct measurements, it allows the rapid gathering of information over system behaviors stretching over long periods of time. We plan therefore to continue to make discrete simulation an integral part of our current study of managment schemes for replicated data objects [PaLo88].

Acknowledgements

The authors wish to thank Walter A. Burkhard and all the other members of the Gemini group for their help and their encouragement. We are grateful to Robb Mills for providing us with the code used to compute batch averages, and we wish to acknowledge the assistance of Keith Muller, Rick Ord, and Greg Hidley in the specification of our model.

References

[BGS86] D. Barbara, H. Garcia-Molina and A. Spauster, "Policies for Dynamic Vote Reassignment," *Proc. Sixth International Conference on Distributed Computing Systems* (1986), pp. 37-44.

[BMP87] W.A. Burkhard, B. E. Martin and J.-F. Pâris, "The *Gemini* Fault-Tolerant File System: the Management of Replicated Files," *Proc. Third International Conference on Data Engineering,* (1987), pp. 441-448.

[CLP87] J.L. Carroll, D. Long and J.-F. Pâris, "Block-Level Consistency of Replicated Files," *Proc. Seventh International Conference on Distributed Computing Systems,* (1987), pp. 146-153.

[DaBu85] D. Davcev and W.A. Burkhard, "Consistency and Recovery Control for Replicated Files," *Proc. Tenth ACM Symposium on Operating System Principles,* (1985), pp. 87-96.

[Elli77] C. A. Ellis, "Consistency and Correctness of Duplicate Database Systems," *Operating Systems Review,* 11, 1977.

[ElFl83] C. S. Ellis and R. A. Floyd, "The Roe File Systems," *Proc. Third Symposium on Reliability in Distributed Software and Database Systems,* (1983).

[Fish78] G. S. Fishman *Principles of Discrete Event Simulation,* New York: Wiley and Sons, Inc., 1978.

[Giff79] D. K. Gifford, "Weighted Voting for Replicated Data," *Proc. Seventh ACM Symposium on Operating System Principles,* (1979), pp. 150-161.

[Good83] N. Goodman, D. Skeen, A. Chan, U. Dayal, R. Fox and D. Ries, "A Recovery Algorithm for a Distributed Database System," *Proc. Second ACM Symposium on Principles of Database Systems,* (1983), pp. 8-15.

[Jajo87] S. Jajodia, "Managing Replicated Files in Partitioned Distributed Database Systems," *Proc. Third International Conference on Data Engineering,* (1987).

[JaMu87] S. Jajodia and D. Mutchler, "Dynamic Voting," *Proc. ACM SIGMOD 1987 Annual Conf.,* San Francisco, Calif. (May 1987), pp. 227-238.

[LaCa79] A. Law and J. Carson, *A Sequential Procedure for Determining the Length of a Steady-State Simulation, Operations Research,* Vol. 27 (1979), 1011-1125.

[LaMi88] A. Law and R. Mills, *Statistical Analysis of Simulation Output Data Using SIMSCRIPT II.5,* Los Angeles: CACI, Inc., 1988.

[LoPa87] D.D.E. Long and J.-F. Pâris, "On Improving the Availability of Replicated Files," *Proc. Sixth Symposium on Reliability in Distributed Systems and Database Systems,* (1987), pp. 77-83.

[NKT87] V. F. Nicola, V. G. Kulkarni and K. S. Trivedi, "Queueing Analysis of Fault-Tolerant Computer Systems," *IEEE Trans. Software Engineering,* Vol. SE-13, No. 3 (March 1987), 363-375.

[PaLo88] J.-F. Pâris and D.D.E. Long, "Efficient Dynamic Voting Algorithms," *Proc. Fourth International Conference on Data Engineering,* Los Angeles, Calif. (February 1988), to appear.

[Pari86] J.-F. Pâris, "Voting with Witnesses: A Consistency Scheme for Replicated Files," *Proc. Sixth International Conference on Distributed Computing Systems,* (1986), pp. 606-612.

[PaBu86] J.-F. Pâris and W. A. Burkhard, "On the Availability of Dynamic Voting Schemes," Computer Science Technical Report, Department of Computer Science and Engineering, University of California, San Diego, 1986.

[PaLo88] J.-F. Pâris and D.E. Long, "Efficient Dynamic Voting Algorithms. *Proc. Fourth International Conference on Data Engineering,* Los Angeles, Calif. (February 1988).

[Pope81] G. Popek, B. Walker, J. Chow, D. Edwards, C. Kline, G. Rudisin and G. Thiel, "LOCUS: A Network Transparent, High Reliability Distributed System," *Proc. Eighth ACM Symposium on Operating System Principles,* (1981), pp. 169-177.

[SkSt83] D. Skeen and M. Stonebraker, "A Formal Model of Crash Recovery in a Distributed Systems," *IEEE Transactions on Software Engineering,* Vol. SE-9, No. 3 (May 1983), 219-228.

[Thom79] R. H.Thomas, "A Majority Consensus Approach to Concurrency Control," *ACM Transactions on Database Systems,* Vol. 4, (1979), 180-209.

[ScSc83] Schlichting, R. D., and F. B. Schneider, "Fail Stop Processors: An Approach to Designing Fault-Tolerant Computing Systems," *ACM Transactions on Computer Systems,* 1983, 222-238.