Voting with Witnesses: A Consistency Scheme for Replicated Files

Jehan-François Pâris[†]

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

International Conference on Distributed Computing Systems, 1986, pages 606–612

Abstract

Voting schemes ensure the consistency of replicated files by disallowing all read and write requests that cannot collect an appropriate quorum of copies. This procedure requires a minimum number of three copies to be of any practical use and tends to disallow a relatively high number of read and write requests.

We propose to replace some of these copies by mere records of the current state of the file. These records, called witnesses, will be assigned weights and participate to the collection of quorums.

We show, that under very general assumptions, the reliability of a replicated file consisting of n copies and m witnesses is the same as the reliability of a replicated file consisting of $n + m$ copies. We also compare the availability of a replicated file consisting of two copies and one witness with that of a file having three copies and show that, under normal circumstances, the two files have similar availabilities.

Keywords: file consistency, distributed file systems, replicated files, voting.

1 Introduction

Various distributed file systems maintain replicated copies of the same file on different hosts [2, 4, 12, 13, 17]. File replication, as this technique is known, has indeed several major advantages: Since file contents are replicated, no single device failure can destroy any data. Storing copies of a file on two or more distinct hosts guarantees that no single host failure will make the file unaccessible. Moreover, access times for read operations are also improved since replication increases the probability that any given read operation can be performed on a local copy of the file.

The existence of several copies of the same file residing on different hosts immediately raises the issue of *file consistency* [1, 3, 9–11, 14, 16, 18]. It would indeed be extremely burdensome for the users of a replicated file system to keep track of the status of every copy of every replicated file. We therefore need a scheme to determine which copies of a the replicated file are up to date. This scheme is to operate correctly in the presence of any combination of host and subnet failures.

Voting is the best known example of such consistency schemes. In its simplest form, voting assumes that the current state of a replicated file is the state of the majority of its copies. Ascertaining the state of a replicated file thus require accessing a majority of its copies. Should this be prevented by one or more failures, the file is considered unavailable.

We present here an extension of voting in which some copies of the file are replaced by much smaller records of the current state of the file. Although not containing any data themselves, these records called witnesses can testify about the current state of the replicated file and can vote like conventional copies.

Section 2 of this paper surveys existing consistency schemes for replicated files. Section 3 introduces witnesses and discusses variants of our basic schemes. Section 4 contains a brief reliability and availability analysis of witness schemes under standard Markovian assumptions. Finally, section 5 has our conclusions.

2 Existing Consistency Algorithms

As pointed out by Gifford [6], algorithms for maintaining replicated data objects fall into two categories. The first ones select for each replicated data object a *primary site,* or *synchronization site,* that performs all update arbitrations. Distributed INGRES [17] and LOCUS [112, 19] follow this approach. The main advantage of the scheme is its simplicity. Its main drawback is its vulnerability to failures of the synchronization site. LOCUS allows then the selection of a new synchronization site; even then all the information present at the former synchronization site is lost.

Algorithms that do not depend on a unique synchronization site are more complex. They can rely either on *queued update*

[†]"Voting with Witnesses: A Consistency Scheme for Replicated Files." In *Proceedings of the* 6^{th} *International Conference on Distributed Computing Systems,* Cambridge: IEEE, 1986, 606–612. Author's address: Department of Computer Science, University of Houston, 501 Philip G. Hoffman Hall, Houston, Texas 77204-3010.

messages or on *voting*. Systems based on queued update messages, like SDD-1 [13,1], rely on the ability of their communication subsystem to buffer update messages over system failures. Voting [3–6, 9, 14, 15, 18], on the other hand, ascertains the current state of a file by polling the accessible copies of the file and attempting to reach a specific quorum. In the simplest case, this quorum is the same for read and write operations and is equal to one copy. Different quorums for read and write operations can be defined and different weights, including none, allocated to every copy. Consistency is guaranteed as long as the two following conditions are met:

1. the write quorum W is high enough to disallow simultaneous writes on two distinct subsets of copies, and
2. the read quorum R is high enough to disallow simultaneous reads and writes on two disjoint sets of copies.

These conditions are simple to verify, which accounts for the conceptual simplicity and the robustness of voting schemes. However, they considerably limit the accessibility of replicated files. Consider, for instance, a replicated file having two physical copies. Should equal weights be assigned to each copy, condition (1) requires that *both* copies be up in order to update the file. As a result, the availability of the replicated file for write access is less than it would have been had the file not been replicated. Should a higher weight be assigned to one of the two copies, conditions (1) and (2) require that this copy be accessible in order to access the file. The existence of a second copy has then absolutely no effect on the reliability or the availability of the file.

A minimum of three copies is thus needed to improve the reliability and the availability of the file for both read and write accesses. The most reasonable solution then is to assign equal weights to all copies of the replicated files and to have both read and write quorums equal to two. The file will then remain accessible as long as two out of the three copies are accessible.

3 Voting with Witnesses

3.1 Motivation

Consider a replicated file with five copies. Assume that all copies are assigned one vote and that the write quorum is equal to three votes. If we attach to each copy of the file a *version number* that is updated every time the copy is updated [6], this version number will give us an unambiguous representation of the state of that copy.

Since our read quorum is equal to three, we need to inspect the version number of three copies to ascertain the state of the replicated file. Thus, if two copies are unaccessible and the version numbers of the three other ones are

(12, 12, 15),

we know that the current version number of the replicated file is 15 and that two out of the three accessible copies are obsolete. To reach this decision, we only had to interrogate the version numbers of the three accessible copies and did not have to examine their contents.

We thus propose to replace some of these copies by the mere recordings of the version number of the file with no data attached to it. We will call these entities *witnesses*. Witnesses are assigned weights like conventional copies and participate like them to the collection of quorums. Whenever a witness happens to belong to a write quorum, its version number is incremented every time the file is updated. Like conventional copies, witnesses are to be stored in stable storage [6, 8]. Witnesses are thus the exact counterparts of the *weak copies* introduced by Gifford, which had data but no weights [6].

Because of their very small sizes, witnesses have practically negligible storage costs. Bringing a witness up to date becomes also a trivial operation since it only involves the update of a version number. Witnesses can thus be created much more freely than conventional copies, the only limiting factor being the message overhead resulting from version number updates at every write.

3.2 Definitions

A *replicated* file consists of a collection of mutually consistent entities having as an ensemble the same operations as a standard file. These entities are distributed over several sites of a computer network. There are two types of entities within replicated files: *copies* and *witnesses*. A copy contains data and a version number that always reflects the most recent write recorded by the copy. Each copy is assigned a specific number of votes which entitles it to participate to all elections involving the replicated file. A witness contains only a version number that always reflects the most recent write recorded by the witness. Each witness is assigned a specific number of votes and is therefore entitled it to participate to all elections involving the replicated file.

3.3 The Collection of Read and Write Quorums

Procedures for collecting read or write quorums are scarcely affected by the presence of witnesses. One can indeed select Gifford's original scheme [6] or one of the several variants that have been developed more recently. In each of these schemes, read and write quorums are to be collected as if the witnesses were conventional copies with the additional restriction that every quorum must include at least one current copy[1].

The restriction expresses the fact that one cannot read from a witness or use it to bring a copy up-to-date. Let us illustrate this by a small example.

[1] Implementors concerned with the possibility of device failures destroying data may decide to disallow writes if the write quorum does not include a minimum of two copies.

Consider a replicated file with three copies and two witnesses. Assume that all copies are assigned one vote and that both read and write quorums are set to three votes. If two entities are unaccessible and the version numbers of the three other ones are 12, 12 and 15, the version numbers of the two unaccessible entities cannot be lower than 15 as update 15 must have involved at least three entities, and cannot be higher than 15 since any subsequent update would have involved more than two copies. We could represent the possible configurations for the three accessible entities as

(12, 12, 15),
(12, 12, 15w),
(12, 12w, 15),
(12, 12w, 15w),
(12w, 12, 15),
(12w, 12, 15w),
(12w, 12w, 15).

where witnesses are indicated by a "w" appended to their version numbers. In three of these seven possible configurations, namely,

(12, 12, 15w),
(12, 12w, 15w),
(12w, 12, 15w),

the only up-to-date entity is a witness and no quorum will be reached. There will thus be circumstances where a replicated file with three copies and two witnesses will not be available while a file consisting of five copies would have been. The chain of events susceptible to such a situation is rather complex. In the previous example, it required that the two entities currently unaccessible became unaccessible after update 15 but before the two entities that missed writes 13, 14 and 15 could be brought up to date. As we will see in section 4, we have strong reasons to conjecture that these situations will remain rare occurrences under most normal circumstances and will have a minimal impact on the reliability and the availability of replicated files with witnesses.

The performance of replicated files can be significantly increased by including in the write quorums additional copies of the file. In the case of a replicated file consisting of three copies, one may wish, for instance, to update all the three copies whenever possible rather than the two copies required to form a write quorum. Since one may expect copies to be accessible most of the time, the average number of copy updates per write will thus be closer to three than to two. The obvious price to pay is an increase in the number of messages being transmitted. No such benefits can be obtained by adding additional witnesses in write quorums. Thus a replicated file consisting of two copies and one witness will never require more than two copy updates per write while a replicated file consisting of three copies will normally require approximately three copy updates per request.

3.4 Extensions to the Basic Scheme

We have presented a consistency scheme where some copies of a replicated file were replaced by mere records of the current version number of the file while keeping their existing weights. An obvious enhancement to the scheme is to make it dynamic and let witnesses be upgraded to copies and copies transformed into witnesses according to the circumstances.

3.4.1 Upgradable Witnesses

Returning to our previous example, let us consider the case where two of the three copies of the replicated file are unaccessible while one copy of the file and two witnesses are accessible. Say, for instance. that the accessible entities are in the configuration (12w, 12w 15). The file remains available and could be updated, which would leave the two accessible witnesses and the the only accessible copy in the configuration (16w, 16w, 16). This might be considered as a less than desirable situation since the file has now only one copy that is kept up to date. Should the expected repair times of the faulting hosts significantly exceed the time required to make an additional copy of the file, one may wish to upgrade one or two of the two accessible witnesses by transforming them into conventional copies. This can be done without altering in any way the consistency scheme as long as the upgraded witnesses keep their weights.

When the two unaccessible copies recover, one will have too many copies and not enough witnesses. This situation will have to be corrected either by returning each entity to its original state or, more simply, by transforming into witnesses some of the newly recovered copies. This latter solution has however the inconvenience of modifying the original lay-out of the replicated file, which may sometimes have an adverse effect on access times.

3.4.2 Temporary Witnesses

Even with conventional voting schemes, one may encounter situations where the collection of a write quorum is impeded by the presence within this quorum of obsolete copies [6]. Conventional voting algorithms require that these obsolete copies be brought up to date before the write request is processed. The resulting delays may be incompatible with the constraints of some real-time applications.

A faster solution would consist of transforming these obsolete copies into temporary witnesses that could be quickly brought up to date by modifying their version numbers. This would avoid any further delays for the write request, which can now be processed immediately. The newly created temporary witnesses will of course have to be brought back as soon as possible to the state of full copies to minimize the impact of the procedure on the reliability and the availability of the replicated file.

3.4.3 Application to File Migration

Gifford had already mentioned that weak copies could be used to bring additional copies of the files to the hosts where the file was currently consulted [6]. Witnesses also allow the removal of copies from hosts that are overloaded or too far away by transforming these copies into witnesses.

4 Reliability and Availability of Witness Schemes

In the section, we will consider a network consisting of several *hosts* linked by a communication subnet. Since this will be the most frequent case, we will assume that copies of the same replicated file will always reside on different hosts. Hosts are subject to failures; these failures may either involve the host itself or its communication interface. When a host fails, a repair process is immediately initiated. This repair process will never fail although it may take any arbitrary amount of time before completing. Should several hosts fail, the repair process will be performed in parallel on these failed hosts. We will also assume that the repair process will attempt to bring up to date all the copies that might have become obsolete during the time the host under repair was not operational. Such attempts will not be always successful since they depend on the availability of up-to-date copies of the replicated file.

We will assume that individual site failures are independent events distributed according to the same Poisson law. In other words, the probability that a given site will experience no failure during a time interval of duration t will be given by

$$e^{-\lambda t}$$

where λ is the *failure rate.* Similarly, we will require that individual site repairs also be independent events distributed according to a same Poisson law. The probability that a given site will be repaired in Jess than t time units will be given by

$$1 - e^{-\mu t}$$

where μ is the *repair rate.*

Although the assumption of a constant failure rate λ is usually reasonable, the assumption of exponential repair times is harder to defend on general grounds. Both assumptions are however necessary to represent our system by a Markov process with finitely many states. Although a number of queuing theory problems can now be solved for non-exponential distributions, similar methods are not used in reliability theory as they would lead to "cumbersome formulas that are unsuitable for computation" [7].

4.1 Reliability of Replicated Files with Witnesses

One defines the reliability of a system as the probability that the system will operate correctly over a time interval of duration t given that all of its units were operating correctly at time $t = 0$ [7]. The correct operation of a replicated file managed by a voting scheme is guaranteed as long as a sufficient number of copies and witnesses are maintained continuously accessible and at least one of these entities is a copy.

Consider, for instance, the case of a replicated file consisting of three copies. Let us neglect for the moment subnet failures. The file will remain available as long as a quorum of copies can be collected, which will be the case as long as at least two out of the three copies remain continuously accessible. Similarly, a file consisting of of two copies and one witness would also remain available as long as at least two of these three entities remained accessible since

1. two entities are enough to constitute a quorum,
2. one of these two entities will necessarily be a copy, and
3. this copy will necessarily be up to date since the continuous existence of a minimum of two accessible entities guarantees that all copies residing on hosts that have failed will be automatically be brought up to date by the repair process.

The conditions for the continuous operation of a replicated file consisting of two copies and one witness are thus the same as the conditions for the continuous operation of a file consisting of three copies. In other words, replacing one of the three copies by a witness does not decrease the reliability of the replicated file as long as the failure rate and the repair rate of the host on which the witness resides remains unchanged.

This property will hold for replicated files with an arbitrary number of copies and witnesses as long as the number and the weights of the witnesses do not allow the collection of a quorum entirely made of witnesses. We have thus the following theorem.

THEOREM

The reliability of a replicated file consisting of n copies with weights $w_1, w_2, \ldots, w_n$ and m witnesses with weights $w'_1, w'_2, \ldots, w'_m$ is equal to the reliability of a replicated file consisting of $n + m$ copies with weights $w_1, w_2, \ldots, w_n, w'_1, w'_2, \ldots, w'_m$ provided that that the Dumber and weights of the witnesses do not allow the collection of a quorum entirely made of witnesses.

Proof: Consider a replicated file F consisting of n copies with weights $w_1, w_2, \ldots, w_n$ and m witnesses with weights $w'_1, w'_2, \ldots, w'_m$ and a replicated file G obtained from F by replacing the m witnesses by full copies while keeping their weights. Assume that the two files go through the same pattern of failures and repair. At any time, F and G will have the same number of entities up. As long as G will remain continuously available,

1. F will have enough available entities to constitute a quorum,
2. at least one of these entities will be a copy, and

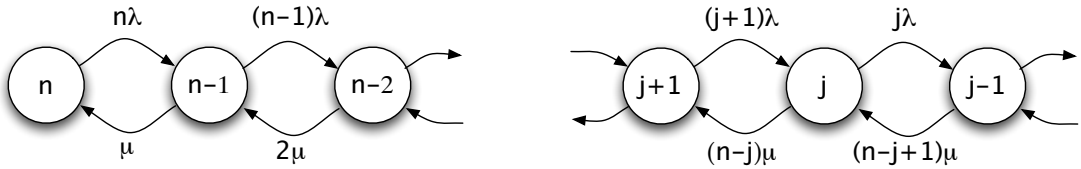

Figure 1: *State-Transition-Rate Diagram for n Copies*

3. this copy will necessarily be up to date since the continuous existence of a valid quorum guarantees that all copies residing on hosts that have failed will be automatically be brought up to date by the repair process. F will thus remain available as long as G does not fail and the two files will have the same availability.

4.2 Availability of a Replicated Files with Witnesses

The availability A of a system is the limiting value of the probability $p(t)$ that that system will be operating correctly at the instant t [7].

$$A = \lim_{t \to \infty} p(t)$$

To simplify our analysis, we will continue to assume that the subnet linking the several hosts on which physical copies of replicated files reside cannot fail. Although not correctly representing the actual behavior of typical point-to-point networks, this hypothesis is an acceptable first-order approximation of the behavior of many carrier-sense local area networks.

Consider first a replicated file consisting of n copies and *no* witnesses. The *state* of this file can then be conveniently represented by the current number of copies that are available at any time. Failures and repairs of sites are the only events that can change the state of the system. We assume that the probability of two such events occurring simultaneously is negligible. Suppose that the system was initially in the state n where all current copies of the file were available. This state will be left for the state $n-1$ if one of the n sites fails. Since each site failure is an independent event, the total rate at which state n will be left is equal to n times the individual failure rate λ of a single site. The state $n-1$ will in turn be left if either (a) the only site that was down is repaired and the system returns to the state n, or (b) one of the $n-1$ sites that were operational now fails.

The rate at which the first event may occur is the repair rate of a single site μ. The rate at which the second event may occur is equal to $n-1$ times the individual failure rate λ of a single site.

In general, the rate at which a state j, with $0 < j < n$, may be left will be given by $j\lambda+(n-j)\mu$, as shown on Figure 1. Writing the equilibrium conditions for our system, one can easily derive the equilibrium state probabilities $p_n, \ldots, p_0$. They are given by

$$p_n = \frac{1}{(1+\rho)^n}$$
$$\ldots$$
$$p_j = \frac{\binom{n}{n-j}\rho^{n-j}}{(1+\rho)^n}$$
$$\ldots$$
$$p_0 = \frac{\rho^n}{(1+\rho)^n}$$

where $\rho = \frac{\lambda}{\mu}$ is the ratio of the failure rate over the repair rate.

If we have an odd number of copies all with equal weights, the availability $A(n)$of the file will be given by

$$A(n) = \sum_{j=n}^{\lceil n/2 \rceil} \frac{\binom{n}{n-j}\rho^{n-j}}{(1+\rho)^n}. \qquad (n \text{ odd})$$

Should there be an even number of copies, we will have to adjust the weights of the copies in order to break ties. What ever scheme we choose, the best that we can do is to allow. access in one half of these ties. The availability of the file will then be given by

$$A(n) = \sum_{j=n}^{n/2+1} \frac{\binom{n}{n-j}\rho^{n-j}}{(1+\rho)^n} + \frac{\binom{n}{n/2}\rho^{n/2}}{2(1+\rho)^n}. \qquad (n \text{ even})$$

In particular, we have

$$A(2) = \frac{1}{1+\rho} = A(1)$$

and

$$A(3) = \frac{1+3\rho}{(1+\rho)^3}.$$

Since voting with two copies requires one particular copy to be up in order to access the file, the availability of replicated file having two physical copies managed by a traditional voting scheme is no better than the availability of a standard nonreplicated file and we have $A(2) = A(1)$.

The analysis of the availability of a replicated file with witnesses is somewhat more complex. First, we will have to distinguish between witness and complete copies. Secondly, we will

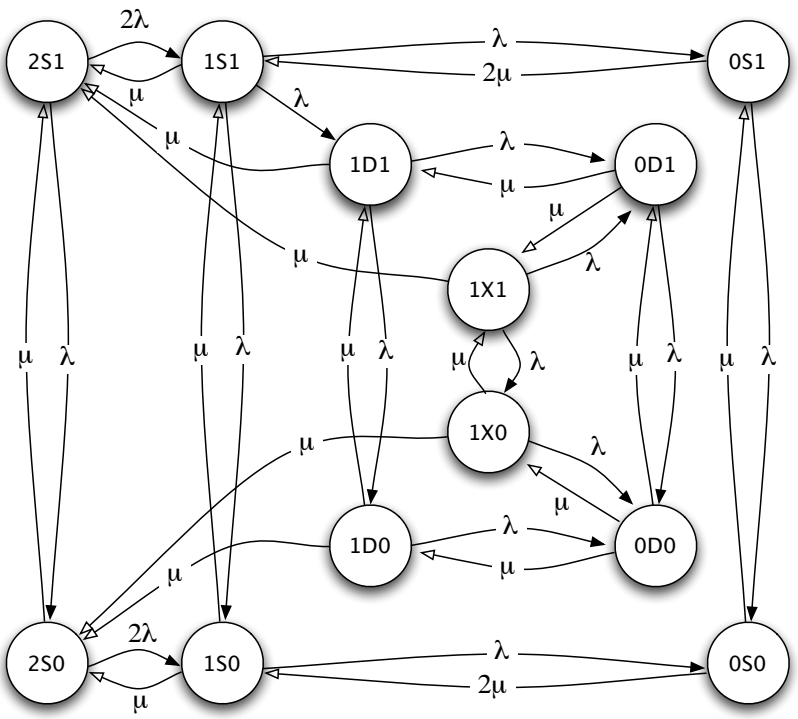

Figure 2: *State-Transition-Rate Diagram for Two Complete Copies and One Witness*

have to take into account situations where the witnesses and the copies disagree about the state of the file. Because of this additional complexity, we will confine our analysis to the case of a replicated file having *two* complete copies and *one* witness.

The state of the system will be represented by two numbers separated by a letter. The first of these two numbers will represent the number of available copies, namely 2, 1 or 0. The second number will represent the current status of the witness, namely up (1) or down (0). The middle letter will represent the possible statuses of the two copies: They can have the same version number (S), have different version numbers (D) or have different numbers and the obsolete copy up and the current copy down (X). Not all possible combinations of numbers and letter are possible. For instance, when the two copies are simultaneously available, they necessarily have the same version numbers. We have thus the two states "2S1" and "2S0." Similarly the letter "X" only appears when one of the copies is up and the other down. Transitions between these states are represented on Figure 2. They obey the following rules:

1. One or more failure transitions originate from every state having at least one copy or one witness up. The rates of the transitions are proportional to the number of copies or witnesses that are up and therefore susceptible to fail. For instance, state "2S1" corresponds to the case where both copies and the witness are up. State "2S1" therefore has one failure transition to state "1S1" with rate 2λ, which corresponds to the failure of either one of the two copies, and one failure transition to state "2S0" with rate λ, which corresponds to the failure of the witness.

2. One or more repair transitions originate from every state having at least one copy or one witness down. The rates of the transitions are proportional to the number of copies or witnesses that are down and therefore susceptible to be repaired (with the only exception of the transitions originating from the states "0D1" and "0D0"). For instance, state "1D1" has one copy down and therefore has one repair transition to state "2S1" corresponding to the repair of that copy.

3. When the file has both copies unavailable and in different states (as in states "0D1" and "0D0"), there is an equal chance that the first copy to be repaired would he the current one or the obsolete one. Thus state "0D1" has one repair transition with rate μ to state "1D1" and one repair transition with rate μ to state "1X1" while state "0D0" has similar repair transition to states "1D0" and "1X0".

4. The only possible transition from a "s" state to a "D" state is from state "1S1" to state "1D1." It occurs when the file is updated. The rate at which this transition occurs is thus given by the rate v at which the replicated file is being updated.

The equilibrium conditions for our system are thus given by

$$\begin{aligned}
p_{2S1}3\lambda &= \mu(p_{2S0} + p_{1X1} + p_{1D1} + p_{1S1}) \\
p_{2S0}(2\lambda + \mu) &= p_{2S1}\lambda + \mu(p_{1X0} + p_{1D0} + p_{1S0}) \\
p_{1S1}(2\lambda + \nu + \mu) &= 2p_{2S1}\lambda + \mu(p_{1S0} + 2p_{0S1}) \\
p_{1D1}(2\lambda + \mu) &= \nu p_{1S1} + \mu(p_{1D0} + p_{0D1}) \\
p_{1X1}(2\lambda + \mu) &= \mu(p_{1X0} + p_{0D1}) \\
p_{1S0}(\lambda + 2\mu) &= (2p_{2S0} + p_{1S1})\lambda + 2\mu p_{0S0} \\
p_{1D0}(\lambda + 2\mu) &= p_{1D1}\lambda + \mu p_{0D0} \\
p_{1X0}(\lambda + 2\mu) &= p_{1X1}\lambda + \mu p_{0D0} \\
p_{0S1}(\lambda + 2\mu) &= p_{1S1}\lambda + \mu p_{0S0} \\
p_{0D1}(\lambda + 2\mu) &= (p_{1X1} + p_{1D1})\lambda + \mu p_{0D0} \\
3\mu p_{0S0} &= (p_{1S0} + p_{0S1})\lambda \\
3\mu p_{0D0} &= (p_{1X0} + p_{1D0} + p_{0D1})\lambda
\end{aligned}$$

where $p_{2S1}, \ldots, p_{0D0}$ are the equilibrium probabilities for all twelve states.

Observing that $\sum_j p_j = 1$, one can obtain the solution

$$\begin{aligned}
p_{2S1} &= \frac{1}{(\rho+1)^3} \\
p_{2S0} &= \frac{\rho}{(\rho+1)^3} \\
p_{1S1} &= \frac{4\rho(\rho(\rho+3)+3)}{(\rho+1)^3(\rho(2\rho+3\psi+6)+6(\psi+1))} \\
p_{1D1} &= \frac{\rho(\rho+2)(\rho(4\rho+9)+6)\psi}{(\rho+1)^5(\rho(2\rho+3\psi+6)+6(\psi+1))} \\
p_{1X1} &= \cdots \\
&\cdots
\end{aligned}$$

where $\rho = \lambda/\mu$ $\psi = \nu/\mu$.

The *availability* $A(2,1)$ of the file will be given by

$$\begin{aligned}
A(2,1) &= p_{2S1} + p_{2S0} + p_{1S1} + p_{1D1} \\
&= \frac{7\rho^3 + 18\rho^2 + 15\rho + 3}{3(\rho+1)^5} \\
&+ \frac{2\left(2\rho^5 + 9\rho^4 + 15\rho^3 + 9\rho^2\right)}{3(\rho+1)^5\left(2\rho^2 + 3\rho\psi + 6\rho + 6\psi + 6\right)}
\end{aligned}$$

Observing that this expression is a monotonously decreasing function of ρ and ψ, a lower bound for $A(2,1)$ can be obtained by computing the limit of $A(2,1)$ for ψ going to infinity, i.e. when the file update rate becomes unbounded. This lower bound will be given by

$$L(2,1) = \frac{7\rho^3 + 18\rho^2 + 15\rho + 3}{3(\rho+1)^5}.$$

Figure 3 contains the compared availabilities of replicated files with one or two copies, three copies and two copies and a witness for values of ρ between 0 and 0.2 and ψ equal to infinity. Replicated files consisting of two copies and one witness that are less often updated will have availabilities falling in the area between those of files with three copies and files with two copies and one witness being constantly updated. As one can see, the availability of a replicated file with two copies and one witness remains very close to the availability of a replicated file with three copies as long as ρ remains small. This will be the case in all non-pathological systems, since we may expect sites to be repaired at a much faster rate than they break down.

5 Conclusions

We have presented a novel consistency scheme for replicated files based on the concept of witness. Witnesses are mere records of the current state of the file. They are assigned weights like conventional copies and participate like them to the collection of quorums. Since they contain no data about the contents of the file, they occupy a negligible amount of secondary storage Unlike conventional voting schemes that required a minimum number of three copies to be of any practical use, our scheme can operate efficiently with two copies and *one* witness.

We have shown under very general assumptions that the reliability of a replicated file consisting of n copies and m witnesses is the same as the reliability of a replicated file consisting of $n+m$ copies. A comparison of the availability of a replicated file consisting of two copies and one witness with that of a file having three copies also showed that, under normal circumstances, the two files have comparable availabilities.

We are currently studying the performance of replicated files consisting of more than two copies and one witness. Our first results are very encouraging although we have still to derive analytical models for more than two copies. Another area still under investigation concerns the feasibility of dynamic schemes allowing witnesses to be upgraded to copies and copies transformed into witnesses.

Witnesses will be included in the second version of the fault-tolerant file system *Gemini* currently under development at the University of California, San Diego. We found that they significantly reduce the cost of managing replicated files and thus hope that they might be a key factor of a greater acceptance of file replication by the UNIX user community.

Acknowledgments

I wish to thank Walter Burkhard, Bruce Martin and all the members of the Gemini group for their help and their encouragements; Darrell Long and Melanie Singer deserve special thanks for their careful implementation of witnesses. The Markov analysis of the availability of witness schemes has been done with the aid of MACSYMA, a large symbolic manipulation program developed at the Massachusetts Institute of Technology Laboratory for Computer Science and supported from 1975 to 1983 by the National Aeronautics and Space Administration under grant

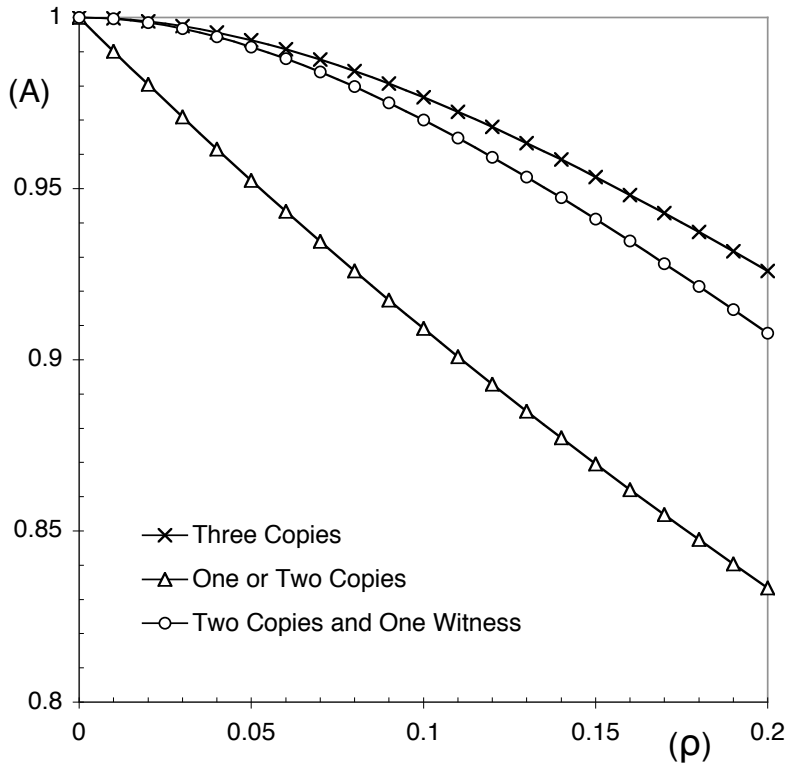

Figure 3: *State-Transition-Rate Diagram for Two Complete Copies and One Witness*

NSG 1323, by the Office of Naval Research under grant N00014-77-C-0641, by the U. S. Department of Energy under grant ET-78-C-02-4686, and by the U. S. Air Force under grant F49620-79-C-020, and since 1982 by Symbolics, Inc. MACSYMA is a trademark of Symbolics, Inc.

References

[1] P. A. Bernstein and D. W. Shipman, "The Correctness of Concurrency Control Mechanisms in a Systems for Distributed Databases–(SDD-1)," *ACM Transactions on Database Systems,* 5, 1980.

[2] W. A. Burkhard, B. E. Martin and J.-F. Paris, "The Gemini Fault-Tolerant File System: the Management of Replicated Files," Computer Sciences Technical Report, Department of EE&CS, University of California, San Diego, 1985.

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

[4] C. A. Ellis and R A. Floyd, "The Roe File Systems," *Proceedings Third Symposium on Reliability in Distributed Software and Database Systems,* 1983.

[5] H. Garcia-Molina, "Elections in a Distributed Computer System," *IEEE Transactions on Computers,* C-31, 1982, pp. 48–59.

[6] D. K. Gifford, "Weighted Voting for Replicated Data," *Proceedings Seventh ACM Symposium on Operating System Principles,* 1979, pp. 150–161.

[7] B. V. Gnedenko, Yu. K. Belyayev and A. D. Soloviev, *Mathematical Methods of Reliability Theory,* (English translation of "Matematicheskiye Metody V Teorii Nadezhnosti"), Academic Press, New York, 1969.

[8] B. W. Lampson, *Distributed Systems – Architecture and Implementation,* Springer-Verlag, New York, 1983.

[9] D. A. Mensace, G. J. Popek, R. A. Muntz, "A Locking Protocol for Resource Coordination in Distributed Databases," *ACM Transactions on Database Systems,* 5, 1980.

[10] T. Minoura and G. Wiederhold, "Resilient Extended True-Copy Token Scheme for a Distributed Database Systems," *IEEE Transactions on Software Engineering,* SE-8, 1982, pp. 173–189.

[11] D. S. Parker *et al.*, "Detection of Mutual Inconsistency on Distributed Systems," *IEEE Transactions on Software Engineering,* SE-9, 1983.

[12] G. Popek, B. Walker, J. Chow, D. Edwards, C. Kline, G. Rudisin and G. Thiel, "LOCUS: A Network Transparent, High Reliability Distributed System," *Proceedings Eight ACM Symposium on Operating Systems Principles,* Pacific Grove, 1981, pp. 169–177.

[13] J. B. Rothnie, N. Goodman and P. A. Bernstein. "The Redundant Update Methodology of SDD-1: A System for Distributed Databases (The Fully Redundant Case)," Report No. CCA-77-02, Computer Corporation of America, 1977.

[14] J. Seguin, G. Sergeant, and P. Wilms, "A Majority Consensus Algorithm for the Consistency of Duplicated and Distributed Information," *Proceedings First International Conference on Distributed Computing Systems,* 1979, pp. 617–624.

[15] D. Skeen, "A Quorum-Based Commit Protocol," *Proceedings Sixth Berkeley Workshop on Distributed Data Management and Computer Networks,* February 1982, pp. 69–80.

[16] D. Skeen and M. Stonebraker, "A Formal Model of Crash Recovery in a Distributed Systems," *IEEE Transactions on Software Engineering,* SE-9, 3, (May 1983), pp. 219–228.

[17] M. Stonebraker, "Concurrency Control and Consistency of Multiple Copies of Data in Distributed INGRES," *IEEE Transactions on Software Engineering,* SE-5, 3 (May 1979), pp. 188–194.

[18] R. H. Thomas, "A Majority Consensus Approach to Concurrency Control," *ACM Transactions on Database Systems,* 4, 1979, pp. 180–209.

[19] B. Walker, G. Popek, R. English, C. Kline and G. Thiel, "The LOCUS Distributed Operating System," *Proceedings Ninth ACM Symposium on Operating Systems Principles,* Bretton Woods, 1983, pp. 49-70.