## Title

The performance of available copy protocols for the management of replicated data

**Permalink**
https://escholarship.org/uc/item/8gp6t7zj

**Journal**
Performance Evaluation, 11(1)

**ISSN**
0166-5316

**Authors**
Pâris, Jehan-François
Long, Darrell DE

**Publication Date**
1990-04-01

**DOI**
10.1016/0166-5316(90)90024-d

Peer reviewed

# The Performance of Available Copy Protocols for the Management of Replicated Data[†]

Jehan-François Pâris
Department of Computer Science
University of Houston
Houston, TX 77204-3475, USA

Darrell D. E. Long
Computer and Information Sciences
University of California
Santa Cruz, CA, 95064 USA

*Performance Evaluation, vol. 11, no. 1, April 1990, pp. 9–30.*

**Abstract**

Available copy protocols guarantee the consistency of replicated data objects against any combination of non-Byzantine failures that do not result in partial communication failures. While the original available copy protocol assumed instantaneous detection of failures and instantaneous propagation of this information, more realistic protocols that do not rely on these assumptions have been devised. Two such protocols are investigated in this paper: a *naive available copy* (NAC) protocol that does not maintain any state information, and an *optimistic available copy* (OAC) protocol that only maintains state information at write and recovery times. Markov models are used to compare the performance of these two protocols with that of the original available copy protocol. These protocols are shown to perform nearly as well as the original available copy protocol, which is shown to perform much better than quorum consensus protocols.

**Keywords**: Mutual Consistency, Fault-Tolerant Systems, Replicated Data, Available Copy, Majority Consensus Voting, Dynamic Voting.

## 1 Introduction

Critical data are often replicated in distributed systems to reduce their read access times or increase their availability in the presence of system failures [4, 5, 28, 34]. A major issue in the management of replicated data is the choice of the *consistency protocol* used to guarantee that all users share the same view of the replicated data objects. Several protocols have been proposed, including *majority consensus voting* [10, 33], *general quorum consensus* [13], *voting with witnesses* [24, 23], *dynamic voting* [7, 14], *available copy* [2, 11, 17], and the *regeneration algorithm* [29].

Until recently, few studies have been dedicated to the performance of these protocols, either in terms of message traffic overhead or in terms of the availability and reliability of the replicated data managed by such protocols. As a result, investigations of consistency protocols have often focused on algorithms with relatively poor fault-tolerance characteristics such as majority consensus voting while more resilient protocols such as available copy or dynamic voting have received considerably less attention.

Available copy protocols were designed to provide higher data availabilities and reliabilities than voting protocols in environments that preclude partial communication failures. Since they discount the possibility of network partitions, available copy protocols can allow access to a replicated data object as long as a single replica of the data object remains available. As a consequence, replication with available copy protocols is a viable technique with two replicas while voting protocols require at least three replicas to garner any improvement in availability over an ordinary unreplicated data object.

[†] Some of the work reported in this paper was performed while the authors were with the Computer Systems Research Group, Department of Computer Science and Engineering, University of California, San Diego. It was supported in part by a grant from the NCR Corporation and the University of California MICRO program. Parts of this paper were presented at the Sixth Symposium on Reliability in Distributed Software and Database Systems and the Seventh International Conference on Distributed Computing Systems.

Several reasons can be advanced to explain why available copy protocols have not yet received the attention that they deserve. First, most earlier computer networks used point-to-point subnets. They were therefore subject to partial communication failures. Second, the original available copy protocol [2, 11] assumed instantaneous detection of site failures. As a result, available copy protocols were often deemed too difficult to implement. Finally, the lack of any comprehensive study of protocol performance impeded the adoption of a more resilient but somewhat more complex protocol.

These considerations do not carry the same weight today. Most local-area networks now use broadcast subnets. Replicated data objects stored on such networks are not subject to network partitions as long as all the sites holding replicas are on the same token ring or CSMA/CD segment. Inexpensive variants of the original available copy protocol have been proposed [3, 6, 17]. None of them require instantaneous detection of site failures. Studies of the performance of available copy protocols have shown that these protocols offer a much higher data availability than majority consensus voting [6, 17] and should be used in combination with the regeneration algorithm to improve its data availability [19, 21].

We present the first comprehensive evaluation of available copy protocols in terms of availability, reliability, mean time to failure and mean time to repair. Our analysis covers the original available copy protocol and two variants that do not require instantaneous detection of site failures. The first, *naïve available copy* maintains no record of site failures; the second, *optimistic available copy* updates this information only at write and recovery time. We compare the performance of these three protocols with the performances of dynamic voting and general quorum consensus protocols. We also compare the message traffic of the naïve and optimistic available copy protocols.

Most of our results are explicitly derived from Markov models of replicated data objects. We found these models well suited to the simple failure modes encountered in networks that cannot partition. Similarly, several previous studies of the availability of dynamic voting protocols [18, 25, 26, 27] have failed to show significant discrepancies between the results obtained from Markov models and discrete event simulation.

In the next section we review the original available copy protocol and present the naïve and optimistic available copy protocols. In Section 3, we present an analysis of the reliability and availability of these three protocols and compare their performance with those of the voting protocols. In Section 4, we compare the message costs of the naïve and optimistic protocols with that of majority consensus voting. Section 5 summarizes our findings.

## 2 Available copy protocols

Available copy protocols are based on the observation that if any one site has been continuously accessible it holds the current version of the data object. Consistency is insured by sending each write to every available copy.

There are three parts to an available copy protocol: write, read and recovery. The rule for writing is extremely simple: *write to all accessible replicas.* Since all accessible replicas receive each write, they are kept in a consistent state: data can then be read from any accessible replica. If there is a replica of the data at the local site, then the read operation can be accomplished locally, avoiding network traffic.

When a site holding a replica recovers from a failure, this replica needs to be compared with another replica that contains the current version of the data object. If all sites holding replicas of the data object have failed, no replica can recover until the last site to fail can be found. The original available copy protocol [2, 11] relied on a complex mechanism to locate that site. Several sets of failure information had to be maintained in real time, including the set of sites participating in the replication of the data object, the set of sites that had been specifically *included* and the set of sites that had been specifically excluded. An *included* site is one that is known to hold a current replica of the data object, an *excluded* site is one that has failed and the failure has been detected by an operational site.

When a site $s$ fails another site $t$ must detect that failure and execute the transaction exclude(s). A failure detection mechanism is assumed both to exist and to be *fool-proof.* When a site $t$ repairs following a failure, it attempts to locate another site $s$ that is operational. If such a site can be found, then $t$ will repair from $s$ and request $s$ to execute the transaction include(t). In the presence of a total failure, the information maintained by the include and exclude transactions is used to determine the last site, or set of sites, to fail. That site is guaranteed to hold a current replica of the data object.

The most controversial assumption is that failures are easily detected and notification of their occurrence can be broadcast to all surviving sites [11]. In general, failures are difficult to detect in a reliable manner. Time-outs are the most common method of detecting failures, but they can delay processing and are unreliable with heavily loaded sites. The original implementation of the available copy protocol required a complex system of monitoring processes for detecting site failure [12].

The two following protocols do not require instantaneous failure detection. The simpler of these, naïve available copy, maintains no system state information. Our other protocol, optimistic available copy, maintains system state information only at write and recovery time. Optimistic available copy approaches the performance of the original protocol since the failure information may be out-of-date, affecting recovery from total failure. But, as the analysis will show, its performance is nearly indistinguishable from that of the original protocol for typical access rates.

## 2.1 Naïve available copy

Naïve available copy does not maintain any site failure information. It behaves like the other available copy protocols except in the event of a total failure, in which case it must wait for all sites participating in the replication to recover. Our experience indicates that total failures seldom occur in practice, and when they do it is usually due to some catastrophic event such as a power failure. A protocol implementing instantaneous detection of failures would not perform better as it would record a simultaneous failure of all sites holding replicas.

Because our naïve protocol maintains no information about sites holding replicas of the data object, network traffic is reduced at the cost of introducing poor worst-case behavior. It exhibits worst-case behavior following a total failure, and as the analysis will show, its performance is very good since total failures seldom occur.

The recovery protocol for a naïve available copy protocol is presented below.

**Recovery Protocol 2.1.** (1) If a site $t$ recovers from a failure and it finds another site $s$ already available, $t$ can repair from $s$.

(2) If a site $t$ recovers from a failure and it finds no other sites available, $t$ must wait for all other sites to recover. The site $s$ which holds the most recent version of the data is then found by examining the version numbers of all sites.

## 2.2 Optimistic available copy

Our second available copy protocol only changes the availability information when the replicated data object is modified or when a recovery occurs. Our method is called *optimistic* since it operates with system state information that may be out-of-date. Although consistency is not compromised, recovery time increases as the state information ages. The protocol assumes a fixed set of sites connected by a partition-free network that provides a reliably delivered mes- sage service.

The problem of finding the last site to fail has been extensively studied by Skeen [32, 31]. Our protocol maintains two pieces of information: a version number per replicated data object, and a *was-available* set per replica. The *was-available* set for an active replica $s$, denoted $W_s$, lists those replicas that received the most recent change to the data. This includes all replicas that received the most recent write and all replicas that have repaired from $s$ since the last write. The was-available sets can be maintained inexpensively by ascertaining which replicas are operational when the replicated data object is first accessed and by sending this information along with the first write; the second write will contain the set of replicas which received the first write and so forth. By delaying the information in this way, communication costs are minimized at the expense of some increase in recovery time.

Since optimistic available copy operates using out-of-date system state information, it is necessary to compute the closure of the was-available set with respect to the recovering site $s$ in order to find the last site to fail. The *closure* of a was-available set $W_s$, written $C^*(W_s)$, is given by

$$C^*(W_s) = \bigcup_{k=0}^{n} C^k(W_s)$$

where $C^k(W_s) = \bigcup_{t \in W_s} C^{k-1}(W_t)$ and $C^0(W_s) = W_s$.

A site $t$ is said to be a *successor* of a site $s$ if $t$ repaired from $s$ and $s$ subsequently failed. For the purposes of this paper, the transitive closure of the successor relation is considered. Thus, for a set of sites $S = \{s_1, \ldots, s_n\}$, saying that $s_i$ succeeds $s_j$ means that there can be any number of intermediate successors up to $n-2$.

Sites are in one of three states: *failed*, *comatose* or *available*. A failed site is one that has ceased to function due to hardware or software failure. Clean failure is assumed. If a site fails it simply halts. Malevolent failures are not tolerated. We assume that this fail-stop behavior can be approximated by an appropriate software layer [30].

A *comatose* site is one that has been repaired but does not know the current state of the replicated data object. A site enters this state following a total failure and remains there until the current version of the data object can be found by examining the version numbers of the other sites. A site that has been continuously operational or that has recovered is said to be available.

Our optimistic available copy protocol bridges the gap in availability between naïve available copy and the original available copy protocol. It provides better worst-case performance than naïve available copy since it need not wait for all sites to recover following a total failure. As the frequency of write requests increases the system state information becomes more current, resulting in quicker recovery. By modifying the availability information of the sites only when a write or recovery occurs, the amount of network traffic is less than the original protocol. The recovery protocol appears below.

**Recovery Protocol 2.2.** (1) If a site $s$ recovers from a failure and it finds that $W_s = \{s\}$, indicating that $s$ was the last site to fail, $s$ is made immediately available.

(2) If a site $t$ recovers from a failure and it finds another site $s$ already available, $t$ repairs from $s$ and $t$ is then added to the was-available sets of all available replicas.

(3) If a site $t$ recovers from a failure and it finds no other sites available, $t$ must wait for all other sites in $C^*(W_t)$ to recover. A site $s$ that holds the current version of the data object must be in $C^*(W_t)$. Site $t$ repairs from site $s$ and $t$ is then added to the was-available sets of all available replicas.

The following theorem establishes the correctness of our access and recovery protocols. Its proof consists of four parts, considering the cases of write, site failure and the three cases introduced by the recovery protocol. The read operation does not affect the state of the system and so is not considered.

**Theorem 2.3.** *The closure of the was-available set of any site always contains the name of a site that holds a current replica of the data object.*

**Proof.** Assume that the invariant holds, as it does in the initial state where all sites are available and $\forall s \in S, W_s = S$. Consider any configuration where the invariant holds, then there are four ways by which the state of the system can change: a write operation can occur, a site can fail, a site can recover and find a replica already available, or a site $s$ can recover and be required to wait for all sites in $C^*(W_s)$ to recover. Each case is considered separately.

(1) When a write occurs, there are no comatose sites and all sites which are currently available will receive the write request. Since the write protocol provides the set of available sites as the new was-available set, the was-available sets of all available sites become consistent. The was-available set of each active site now contains the names of all available sites. The was-available sets of the failed sites remain unchanged.

(2) When a site failure occurs, those sites which have failed have as their was-available sets the names of the sites which had current replicas of the data object when the site failed. This is trivially true in the case where all sites have failed since for each site $s$ holding a current version of the replicated data object, $s \in W$, and no more writes can occur. If a site $s$ fails, then $W_s$ contains the set of sites which received the last write. Let $t \in W_s$ be any one of those sites. If $t$ subsequently fails and a write occurs following its demise, then some site $u$ which is a successor of $t$ will hold the most recent version of the data, otherwise $t$ holds the most recent version of the data. In either case, $t \in C^*(W_s)$.

(3) Suppose that when site $s$ recovers there is a replica of the data object at site t available. The replica at site $s$ will be repaired from the replica at site $t$ according to the recovery protocol. When a site $s$ recovers from a failure it is included in the was-available sets of all available sites, insuring that the invariant is preserved.

(4) Similarly, when all of the sites in $C^*(W_s)$ have recovered, the site $t$ that holds the most recent version of the data can then be found. At this point, the recovery of site $s$ is accomplished as in the previous case and the invariant is preserved. □

Another possible design would have read operations updating the was-available sets as write operations do, resulting in a somewhat higher availability as reads often outnumber writes. We decided against it since it would have precluded inexpensive local reads and significantly increased network traffic overhead.

# 3 Reliability and availability analysis

In this section we introduce Markov models for the original available copies protocol and its naïve and optimistic variants. We first derive the availability, mean time to failure, and mean time to repair of replicated data objects managed by these three protocols. Then we analyze their reliability as it utilizes some results from the availability analysis of the naïve available copy protocol. We then compare the performance of available copy protocols with that of voting protocols.

We define the *availability* $A$ of a data object as the limiting value of the probability $p(t)$ that the data object remains accessible at time $t$,

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

Some operations on replicated data objects experience different availabilities $A_i$. When this is the case, we define the data object's availability as the average availability of all its operations weighted by their relative frequencies. The mean time to failure MTTF and mean time to repair MTTR are related to system availability by the relation

$$A = \frac{\mathrm{MTTF}}{\mathrm{MTTF} + \mathrm{MTTR}}.$$

While availability, mean time to failure and mean time to repair measure the time averaged robustness of a protocol, its *reliability* estimates the probability a replicated data object managed by that protocol will remain constantly available over a given period of time. We define the data object's *reliability* $R(t)$ to be the probability that it remains accessible over a time interval of duration $t$ given that all of its units were operating correctly at time $t = 0$.

We assume that the data object's $n$ replicas reside on distinct sites of a computer network, and are subject to failures. When a site fails, a repair process is immediately initiated. Should several sites fail, the repair process is performed in parallel on these failed sites. Once a site has been successfully repaired, the protocol attempts to update those replicas that might have become obsolete during the time the site was being repaired. Such attempts do not always succeed since they depend on the availability of the current replicas. Since the available copy protocol does not operate correctly in the presence of partitions, we assume the communications network linking the sites where the replicas reside does not fail.

We assume that individual site failures and individual site repairs are independent events distributed exponentially. The probability that a site does not experience a failure during a time interval $t$ is $e^{-\lambda t}$ where $\lambda$, is the *failure rate*, and the probability that a site is repaired in less than $t$ time units is $1 - e^{-\mu t}$ where $\mu$ is the *repair rate*.

For simplicity, we assume that all sites have equal failure rates $\lambda$ and repair rates $\mu$. Under these conditions, the availability $A$ of any single site is given by

$$A = \frac{\mu}{\lambda + \mu} = \frac{1}{1 + \rho},$$

where $\rho = \lambda / \mu$, and the probability of finding exactly $k$ sites available is

$$s_k = \left( \begin{array}{c} x \\ y \end{array} \right) A^k (1 - A)^{n-k} = \frac{\left( \begin{array}{c} n \\ k \end{array} \right) \rho^{n-k}}{(1 + \rho)^n}.$$

Since a replicated data object cannot be accessed unless one of its replicas is available, the availability of a replicated data object with n identical replicas obeys the inequality

$$A(n) \quad \leq \quad 1 - s_0 \leq 1 - \frac{\rho^n}{(1 + \rho)^n}. \qquad (1)$$

## 3.1 Available copy

A replicated data object with $n$ replicas managed by an available copy protocol with perfect system state information can be described by a Markov model with $2n$ states. The first $n$ states labeled from $\langle 1 \rangle$ to $\langle n \rangle$ represent the data object's states when 1 to $n$ replicas are available. The $n$ states labeled from $\langle 0 \rangle$ to $\langle \overline{n - 1} \rangle$ represent the data object's states when all replicas entered after a total failure when 0 to $n - 1$ replicas, excluding the last replica to fail, have recovered but remain comatose.

As seen in Figure 1, the transitions between states are grouped into two classes: failure transitions and recovery transitions. State $\langle n \rangle$ has only one out-going transition $\langle n \rangle \Rightarrow \langle n - 1 \rangle$ with rate $n\lambda$. This transition corresponds to the failure of any of the $n$ replicas of the data object. All other available states $\langle j \rangle$ have one out-going failure transition $\langle j \rangle \Rightarrow \langle j - 1 \rangle$ with rate $j\lambda$, and one out-going repair transition $\langle j \rangle \Rightarrow \langle j + 1 \rangle$ with rate $(n - j)\mu$. The situation changes once all replicas have failed. The replicated data object begins in state $\langle \bar{0} \rangle$ and returns to state $\langle 1 \rangle$ if the last available copy recovers first. If any of the $n - 1$ other replicas recover first, that replica remains *comatose* and the replicated data object enters state $\langle \bar{1} \rangle$. As a result, state $\langle \bar{0} \rangle$ has one out-going transition $\langle \bar{0} \rangle \Rightarrow \langle 1 \rangle$ with rate $\mu$ and another $\langle \bar{0} \rangle \Rightarrow \langle \bar{1} \rangle$ with rate $(n - 1)\mu$.

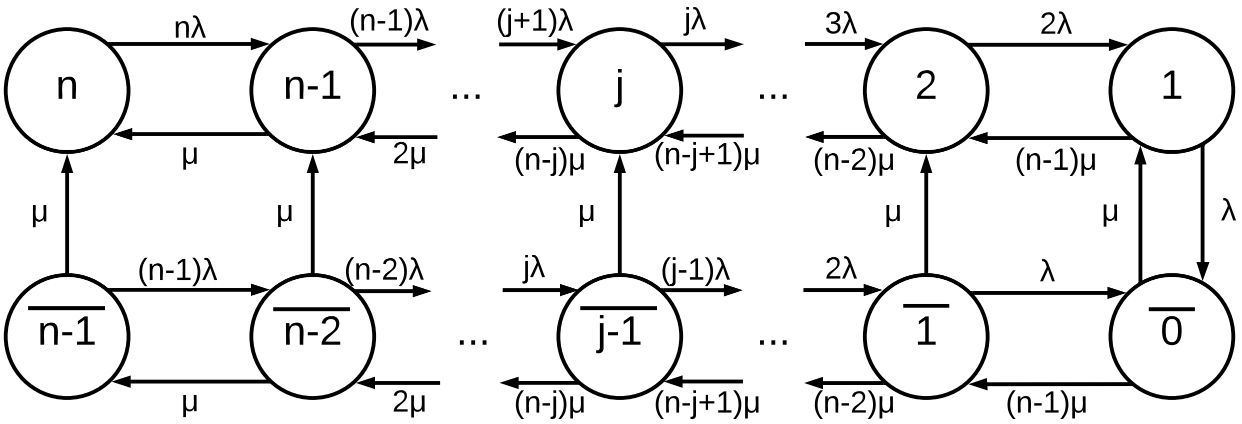

Figure 1: State-transition diagram for available copy.

All states $\langle\bar{j}\rangle$ with $j = 1,\dots,n-2$ have three outward transitions: $\langle\bar{j}\rangle \Rightarrow \langle\overline{j-1}\rangle$ with rate $j\lambda$ corresponding to the failure of the $j$ comatose replicas, another $\langle\bar{j}\rangle \Rightarrow \langle j+1\rangle$ with rate $\mu$ corresponding to the recovery of the last available copy, and a third one $\langle j\rangle \Rightarrow \langle\overline{j+1}\rangle$ with rate $(n-j-1)\mu$ corresponding to the recovery of one of the other $n-j-1$ failed replicas. State $\langle\overline{n-1}\rangle$ lacks a third outward transition since the only failed replica is necessarily the last replica to fail.

Under these condition, the availability $A_{\text{AC}}(n)$ of the replicated data object is

$$A_{\text{AC}}(n) = 1 - \sum_{i=0}^{n-1} \bar{\rho}_i$$

where $\bar{\rho}_i$ denotes the probability that the replicated data object is in the state $\langle\bar{i}\rangle$. These $\bar{\rho}_i$ are linked by the recurrence relation

$$\bar{\rho}_{n-k} = \frac{(n-k+1)\rho}{k-1}\bar{\rho}_{n-k+1} + \frac{1}{k-1}\sum_{j=1}^{k-1}\bar{\rho}_{n-j'}, \quad k > 0$$

where $\rho = \lambda/\mu$, which can be rewritten as

$$\bar{\rho}_i = \frac{C_{n-i-1}}{C_{n-1}}\frac{\rho^n}{(1+\rho)^n}, \quad i = 0,\dots,n$$

with

$$C_0 = 1, \quad C_1 = (n-1)\rho + 1$$

and

$$C_k = \frac{(n-k)\rho}{k+1}C_{k-1} - \frac{n-k+1}{k}C_{k-2} \quad \text{for } k > 1.$$

The availability $A_{\text{AC}}(n)$ of replicated data object with $n$ replicas by the $AC$ protocol is then

$$A_{\text{AC}}(n) = 1 - \sum_{0}^{n-1}\bar{\rho} = 1 - \sum_{i=0}^{n-1}\frac{C_{n-i-1}}{C_{n-1}}\frac{\rho^n}{(1+\rho)^n},$$

which can be rewritten as the quotient of a polynomial of degree $n-1$ in $\rho$ by a polynomial of degree $2n+1$. In particular, we have

$$A_{\text{AC}}(2) = \frac{1+3\rho+\rho^2}{(1+\rho)^3}, \tag{2}$$

$$A_{\text{AC}}(3) = \frac{2+9\rho+17\rho^2+11\rho^3+2\rho^4}{(1+p)^3(2+3\rho+2\rho^2)}, \tag{3}$$

$$A_{\text{AC}}(4) = \frac{6+37\rho+99\rho^2+152\rho^3+124\rho^4+47\rho^5+6\rho^6}{(1+\rho)^4(6+13\rho+11\rho^2+6\rho^3)}. \tag{4}$$

A simple lower bound for the availability can be derived form the equilibrium of flows between states $\langle n\rangle, \langle n-1\rangle, \ldots, \langle 2\rangle, \langle 1\rangle$ and state $\overline{\langle n-1\rangle}, \overline{\langle n-2\rangle}, \ldots, \langle \bar{1}\rangle, \langle \bar{0}\rangle$. Since we have

$$\mu(\bar{\rho}_{n-1} + \bar{\rho}_{n-2} + \cdots + \bar{\rho}_1 + \bar{\rho}_0) = \lambda \rho_1$$

and

$$\rho_1 + \bar{\rho}_1 = n \frac{\rho^{n-1}}{(1+\rho)^n},$$

we can obtain a lower bound for the probability of being in any of the non-available states:

$$\sum_{i=0}^{n-1} \bar{\rho}_i \langle \frac{n\rho^n}{(1+\rho)^n}.$$

Hence,

$$A_{\mathrm{AC}}(n) \quad < \quad 1 - \frac{n\rho^n}{(1+\rho)^n}. \qquad (5)$$

To evaluate the mean time to failure of the replicated data object consider the subset of available states. The average time spent by the data object in this subset of states at every visit is equal to its mean time to fail $\mathrm{MTTF}_{AC}(n)$. By applying Little's Law to that subset, we obtain

$$A_{\mathrm{AC}}(n) = \mathrm{MTTF}_{\mathrm{AC}}(n) \lambda \rho_1.$$

Observing that the overall system failure rate $\lambda\rho_1$ is equal to the overall repair rate $\mu(1-A)$, we have

$$\mathrm{MTTF}_{\mathrm{AC}}(n) = \frac{A_{\mathrm{AC}}(n)}{\mu(1 - A_{\mathrm{AC}}(n))}.$$

The mean time to repair is then

$$\mathrm{MTTR}_{AC}(n) = \frac{1}{\mu}.$$

## 3.2 Naïve available copy

The naïve available copy protocol keeps no record of which replica failed last. Once all the replicas have failed, the recovery protocol waits until all replicas of the data object have recovered. It then selects the replica with the highest version number, marks it as being available and uses it to bring all other replicas up to date.

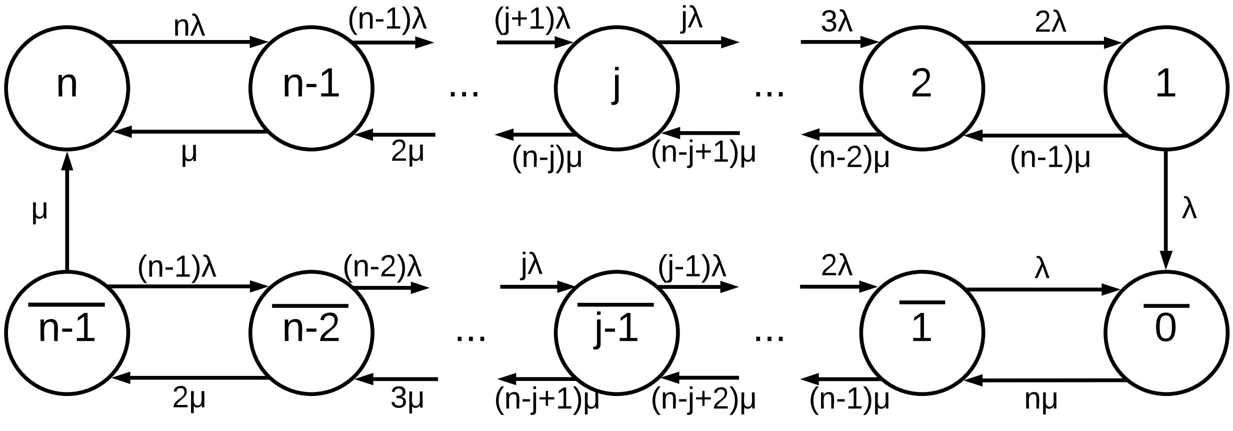

Figure 2: State-transition-rate diagram for naïve available copy.

As seen in Figure 2, the state-transition-rate diagram for a replicated data object with $n$ replicas managed by a naïve available copy protocol has the same $2n$ states as if the data object were managed by an available copy protocol with perfect system state information. Transitions between states are quite similar to those observed for the previous analysis except that there are no transitions from state $\langle j \rangle$ with $j \leq n-2$ to an available state as no information is present to allow early recovery.

From the state transition diagram, we have

$$k\lambda p_k = (n-k+1)\mu p_{k-1} + \lambda p_1 \quad for \;\; k = 2,3,\ldots,n, \tag{6}$$

$$k\mu \bar{p}_{n-k} = (n-k+1)\lambda \bar{p}_{n-k+1} + \mu pn - 1 \quad for \;\; k = 2,3,\ldots,n, \tag{7}$$

$$\lambda p_1 = \mu \bar{p}_{n-1}, \tag{8}$$

from equation (6),

$$p_k = \sum_{j=1}^{k} \frac{(n-j)!(j-1)!}{(n-k)!k!} \rho^{k-j} p_1$$

and from equation (7)

$$\bar{p}_{n-k} = \sum_{j=1}^{k} \frac{(n-j)(j-1)!}{(n-k)!k!} \rho^{k-j} p_{n-1}.$$

The sum of the probabilities of being in any given state must be equal to one,

$$\sum_{k=1}^{n} p_k + \sum_{k=1}^{n} \bar{p}_{n-k} = \sum_{k=1}^{n}\sum_{j=1}^{k} \frac{(n-j)!(j-1)!}{(n-k)!k!} \rho^{j-k} \rho_1 + \sum_{k=1}^{n}\sum_{j=1}^{k} \frac{(n-j)!(j-1)!}{(n-k)!k!} \rho^{k-j} \bar{p}_{n-1} = 1.$$

The expression can be combined with equation (8) to obtain

$$p_1 = \frac{1}{B(n,\rho) + \rho B\left(n, \frac{1}{\rho}\right)}$$

where

$$B(n,\rho) = \sum_{k=1}^{n}\sum_{j=1}^{k} \frac{(n-j)!(j-1)!}{(n-k)!k!} \rho^{j-k}.$$

The availability $A_{\text{NAC}}(n)$ of a replicated data object $n$ replicas managed by a naïve available copy consistency protocol is then given by

$$A_{\text{NAC}}(n) = \sum_{k=1}^{n} p_k = \frac{B(n,\rho)}{B(n,\rho) + \rho B\left(n, \frac{1}{\rho}\right)}.$$

Applying Little's Law to the subset of available states, we have

$$A_{\text{NAC}}(n) = \text{MTTF}_{\text{NAC}}(n)\lambda p_1$$

where $\text{MTTF}_{\text{NAC}}(n)$ is the mean time to fail and $\lambda\rho_1$ is the overall system failure rate.

The former expression can be rewritten as

$$\text{MTTF}_{\text{NAC}}(n) = \frac{A_{\text{NAC}}(n)}{\lambda p_1} = \frac{B(n,\rho)}{\lambda}.$$

The mean time to repair is then given by

$$\text{MTTR}_{\text{NAC}}(n) = \frac{1}{\mu}\frac{1 - A_{\text{NAC}}(n)}{\mu \bar{p}_{n-1}} = \frac{B\left(n, \frac{1}{\rho}\right)}{\mu}.$$

## 3.3 Optimistic available copy

Since optimistic protocols maintain state information only at write requests, their performance is sensitive to the rate at which these requests occur. When write requests are more frequent, the site availability information is closer to the system's true state and availability improves. This is reflected in the analysis where access rates are explicitly considered.

We assume the same set of Markov hypotheses with the addition of the access rate, a Poisson process with mean $\kappa$.

The states of the model are labeled by an ordered triple $\langle i, j, k\rangle$. All unavailable states are marked with a bar, as in $\langle \overline{i, j, k}\rangle$. For each state, $i$ represents the number of available or comatose replicas that belong to the current was-available set, $j$ represents the cardinality of the current was-available set, and $k$ represents the number of comatose replicas that do not belong to the current was-available set.

The state transition diagram for two replicas is illustrated in Figure 3 and for three replicas in Figure 4. In general, the transitions can be grouped into four classes: transitions between available states, transitions from an available state to an unavailable state, transitions between unavailable states, and transitions from an unavailable state to an available state. The reader will note that write transitions, labeled by $k$, can also be used to model the rate at which failures are detected in the original available copy protocol. For each state, the sum of the out-going failure transition rates is equal to $(i+k)\lambda$ and the sum of the rates of all out-going recovery transition rates is equal to $(n-i-k)\mu$.

Among available states there are three types of transitions: failure transitions $\langle i, j, 0\rangle \Rightarrow \langle i-1, j, 0\rangle$ occur with rate $i\lambda$ when $0 < i \leq j$. Recovery transitions $\langle i, j, 0\rangle \Rightarrow \langle i+1, i+1, 0\rangle$ occur with rate $(n-1)\mu$ when $0 < i \leq j$. And, access transitions $\langle i, j, 0\rangle \Rightarrow \langle i, i, 0\rangle$ occur with rate $\kappa$ when $0 < i <\leq j$.

A failure transition from an available state $\langle 1, j, 0\rangle$ to an unavailable state $\langle \overline{0, j, 0}\rangle$ occurs with rate $\lambda$. While recovery transition from an unavailable state $\langle \overline{j-1, j, k}\rangle$ to an available state $\langle j+k, j+k, 0\rangle$ occurs with rate $\mu$ when $k \leq n-j$.

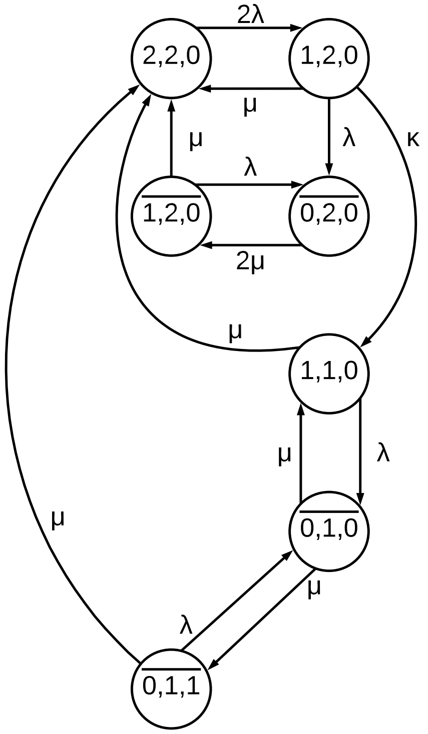

Figure 3: State transition diagram from two optimistic available copies.

Among the unavailable states there are four types of transitions: failure transitions $\langle \overline{i, j, k}\rangle \Rightarrow \langle \overline{i-1, j, k}\rangle$ occur with rate $i\lambda$ when $0 < i < j$ and $k \leq n-j$, and $\langle \overline{i, j, k}\rangle \Rightarrow \langle \overline{i, j, k-1}\rangle$ occur with rate $k\lambda$ when $i < j$ and $0 < k \leq n-j$. Recovery transitions $\langle \overline{i, j, k}\rangle \Rightarrow \langle \overline{i+1, j, k}\rangle$ occur with rate $(j-1)\mu$ when $i < j-1$ and $k \leq n-j$, and $\langle \overline{i, j, k}\rangle \Rightarrow \langle \overline{i, j, k+1}\rangle$ occur with rate $(n-j-k)\mu$ when $i < j$ and $k < n-j$.

The solution to the system of equations for any fixed number of replicas can be found using standard techniques. Symbolic manipulation software is essential because although the process is simple, it is tedious and error-prone.

For example, the availability, $A_{\mathrm{OAC}}(2)$, is given by the sum of probabilities of being in a state where access is

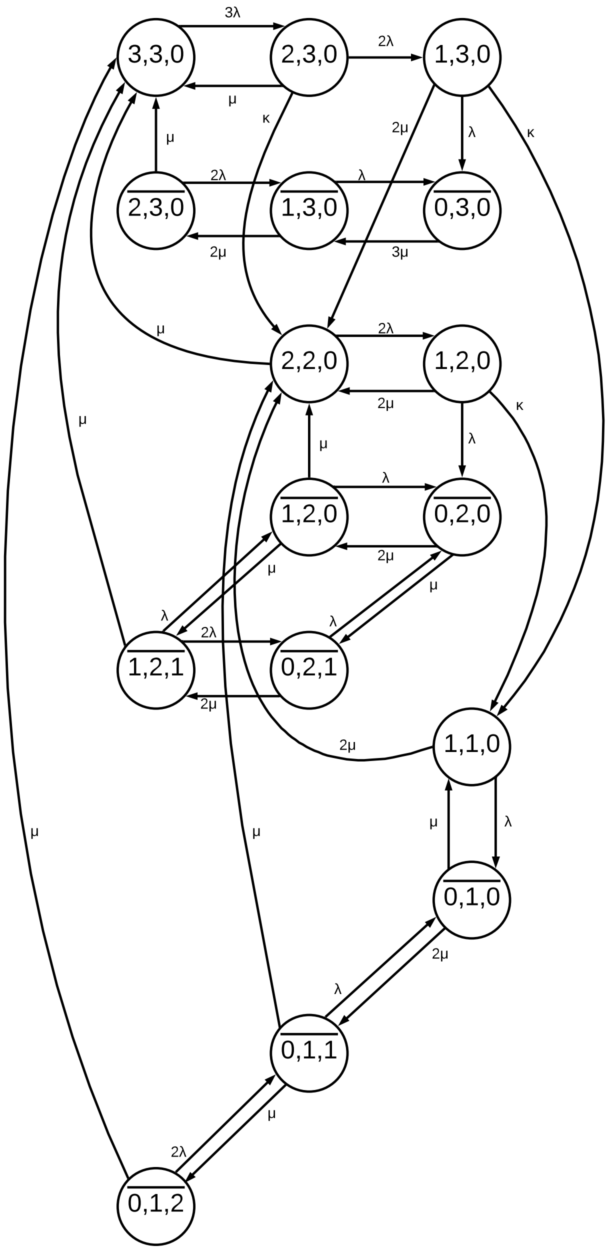

Figure 4: State transition diagram for three optimistic copies.

permitted,

$$A_{\text{OAC}}(2) = p_{\langle 2,2,0\rangle} + p_{\langle 1,2,0\rangle} + p_{\langle 1,1,0\rangle} = \frac{\phi\rho^2 + 3\rho^2 + 3\phi\rho + 4\rho + \phi + 1}{(\rho+1)^3(\rho+\phi+1)}$$

where $\rho = \lambda/\mu$ and $\phi = \kappa/\mu$.

The availability provided by the optimistic available copy protocol approaches the availability provided by an idealized available copy protocol. As the write rate increases, the protocol's state information approaches the system's true state. For $A_{\text{OAC}}(2)$, we have

$$\lim_{\phi\to\infty} A_{\text{OAC}}(2) = \frac{\rho^2 + 3\rho + 1}{(\rho+1)^3} = A_{\text{AC}}(2).$$

In general, the availability afforded by the optimistic available copy protocol, $A_{\text{OAC}}(n)$ approaches the availability provided by an available copy protocol with perfect system configuration information,$A_{\text{AC}}(n)$, as the access rate approaches infinity. This can be seen by considering any of the states with transitions $\langle i,j,0\rangle \Rightarrow \langle i,i,0\rangle$ labeled by $\kappa$. It has been shown [16], that as this transition rate approaches infinity, the probability of the system being in state $\langle i,j,0\rangle$ goes to 0 and the related transition rates are correspondingly increased. The process may be repeated for each transition labeled by $\kappa$. The final result is a system with exactly the same state transitions as an available copy protocol with perfect system configuration information.

For two replicas, optimistic available copy is related to naïve available copy at low access rates. Consider the availability of the data managed by this protocol when accesses are infrequent. For $A_{\text{AOC}}(2)$, we see

$$\lim_{\phi\to 0} A_{\text{AOC}}(2) = \frac{3\rho+1}{(\rho+1)^3} = A_{\text{NAC}}(2).$$

The case where only two replicas are considered is special. To see why a write rate of zero is the same as naïve available copy, consider the state labeled $\langle 1,2,0\rangle$. If the transition labeled by $\kappa$ rate zero, then this transition will never occur and there is no path to state $\langle 1,1,0\rangle$. The resulting diagram is the same as for two naïve available copies.

When a larger number of replicas is considered, this protocol does not degenerate into naïve available copy. The reason is simple: there are paths via the recovery states which lead into the available states. When a recovery occurs, system state information is exchanged and the view of the system becomes up-to-date. As a result

$$\lim_{\phi\to 0} A_{\text{OAC}}(n) > A_{\text{NAC}}(n) \quad \text{for} \quad n > 2.$$

We compare the performance of these protocols for two, three and four replicas in Figure 5, 6, and 7. As expected, the original available copy protocol performs best. But, for reasonable access rates, optimistic available copy quickly converges to a performance level nearly indistinguishable from available copy with instant failure detection. The graphs fail to show significant differences between the three available copy protocols for values of $\rho$ less than 0.10.

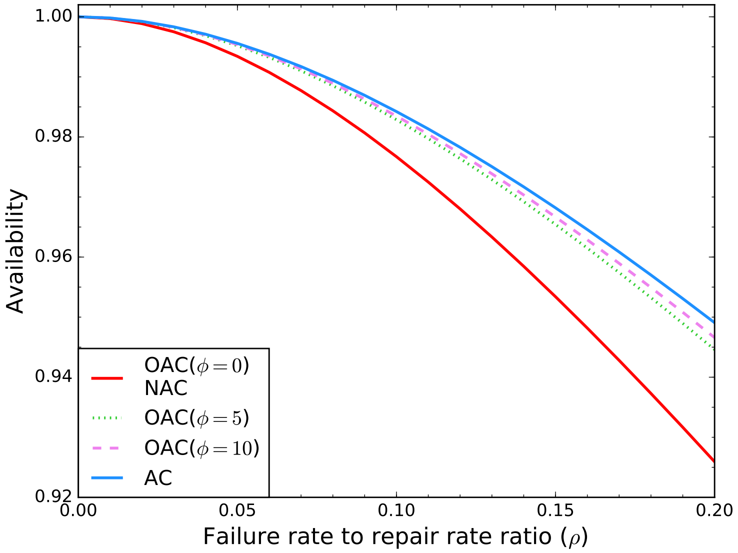

Figure 5: Compared availabilities for two available copies ($\phi$ is the access rate to repair rate ratio).

Modern computers are characterized by availabilities surpassing 0.95 and by values of $\rho$ well below 0.05, suggesting that the naïve protocol performs as well as the original protocol. Observed repair time distributions are characterized by coefficients of variation less than one. Under such conditions, sites tend to recover in the order they failed. The last site to recover after a total failure is often the last one to fail. Under these conditions the original protocol and its optimistic variant will be unable to recover faster than the naïve protocol as they have to wait for the last site to recover in order to find the last current replica of the data object.

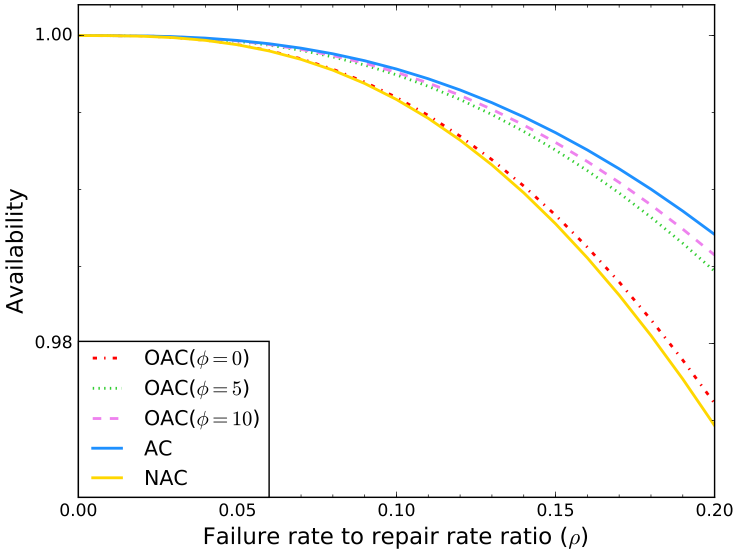

Figure 6: Compared availabilities for three available copies ($\phi$ is the access rate to repair rate ratio).

### 3.4 Reliability analysis

Correct operation of a replicated data object managed by an available copy protocol is guaranteed so long as at least one of the $n$ replicas of the data object remains operational. Thus, replicated data objects managed by available copy, naïve available copy and optimistic available copy protocols have the same reliability $R_{\mathrm{AC}}(n,t)$. This reliability only depends on the number $n$ of replicas and their respective failure and repair rates. Since a replicated data object can only operate when at least one replica is accessible, it follows that consistency protocols that do not generate replicas to replace the ones that failed cannot provide higher reliability than available copy protocols [18].

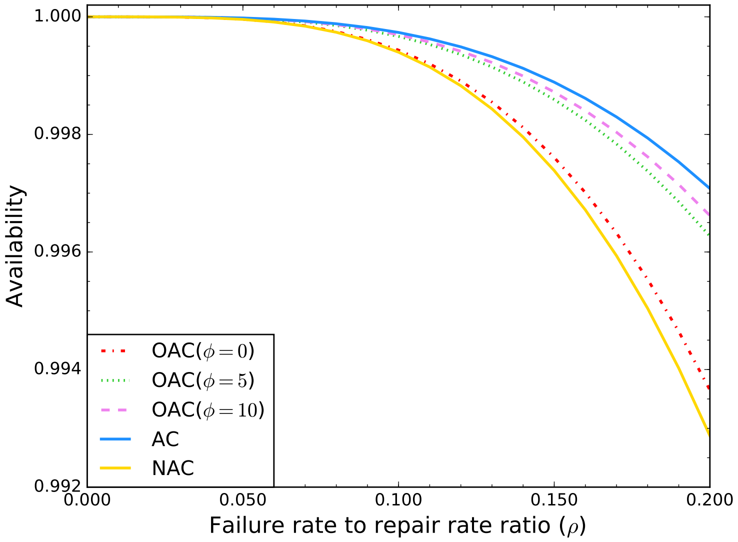

Figure 7: Compared availabilities for four available copies ($\phi$ is the access rate to repair rate ratio).

Systems that remain operational as long as one of a set of $n$ parallel subsystems remains operational are known as 1-*out-of-n* systems. They constitute a special case of $k$-*out-of-n* systems. The evaluation of their reliability requires the solution of $n$ differential equations [9, 20]. McGregor has shown in particular [20] that the reliability of k-out-of-n systems with repairs can be approximated by

$$R(n,t) \approx \exp\left[-\frac{t}{T_m}\right]$$

where $T_m$ is the mean time to failure from an initial configuration where all subsystems are operational. The approximation results in negligible errors for $\mu \geq 5n\lambda$ especially when $n > 3$.

Since $T_m$ is the mean time to fail of the naïve available copy, we have

$$R_{\mathrm{AC}}(n,t) \approx \exp\left[-\frac{\lambda t}{B(n,\rho)}\right].$$

## 3.5 Comparison with voting policies

Voting protocols are the most widely studied class of consistency protocols for replicated data objects, probably due to their simplicity and robustness. In their simplest form, voting protocols assume that the correct state of a replicated data object is the state of the majority of its replicas. Ascertaining the state of a replicated data object requires collecting a quorum of the replicas. Should this be prevented by one or more site failures, the replicated data object is considered to be unavailable.

If there are an odd number of replicas all with equal weights, the *availability* $A_{\mathrm{MCV}}(n)$ of the replicated data object is given by

$$A_{\mathrm{MCV}}(n) = \sum_{j=n}^{\lceil n/2 \rceil} s_j = \sum_{j=n}^{\lceil n/2 \rceil} \frac{\binom{n}{n-j}\rho^{n-j}}{(1+\rho)^n}, \quad n \text{ odd} \tag{9A}$$

If there are an even number of replicas, their weights can be adjusted in order to break the ties occurring when exactly $n/2$ replicas are available. The best that can be done is to allow access in one half of these ties. The availability of the replicated data object is then given by

$$A_{\mathrm{MCV}}(n) = \sum_{j=n}^{n/2+1} s_j + \frac{s_{n/2}}{2} = \sum_{j=n}^{n/2+1} \frac{\binom{n}{n-j}\rho^{n-j}}{(1+\rho)^n} + \frac{\binom{n}{n/2}\rho^{n/2}}{2(1+\rho)^n}, \quad n \text{ even} \tag{9B}$$

which can be rewritten as $A_{\mathrm{MCV}}(2k) = A_{\mathrm{MCV}}(2k-1)$.

Majority consensus voting has been extended to allow for different read and write quorums [10] and to allow non-intersecting write quorums (*general quorum consensus* [13]).

All these protocols are called *static* protocols because the required quorums of replicas and the number of votes assigned to each replica are never modified. Dynamic protocols that adjust quorums, such as dynamic voting and its variants [7, 14, 15, 26], or modify the number of votes assigned to each replica [1] can minimize the impact of site failures. They have been shown to increase the availability over static protocols such as majority consensus voting.

The *dynamic voting protocol* [7] instantly adjusts quorums to reflect changes in the state of the network holding the replicas. The protocol requires each site to maintain in real time a *connection vector* recording the state of the network. Since the original dynamic voting protocol does not assign weights to replicas and does not include a tie-breaking rule, a majority block must always contain at least two replicas. A simple extension, *linear-dynamic voting* [14], resolves ties by applying a total ordering to the sites. *Hybrid dynamic voting* [15], a more recent dynamic protocol, integrates static voting and linear-dynamic voting. Like dynamic voting, it always disallows accesses when less than two replicas are available.

**Theorem 3.1.** The availability $A_{\mathrm{AC}}(n)$ of a replicated data object with $n$ identical replicas managed by the original available copy consistency protocol is greater than the availability $A_{\mathrm{MCV}}(n)$ of a replicated data object with $2n-1$ or $2n$ identical replicas managed by the majority consensus voting protocol as long as the failure rate to repair rate ratio $\rho$ remains less than or equal to one.

**Proof**. Since $A_{\mathrm{MCV}}(2n-1) = A_{\mathrm{MCV}}(2n)$, we only need to prove that $A_{\mathrm{AC}}(n) > A_{\mathrm{MCV}}(2n-1)$ for all $\rho \leq 1$.

- From equations (9A) and (3), we know that $A_{\mathrm{AC}}(2) > A_{\mathrm{MCV}}(3)$ and $A_{\mathrm{AC}}(3) > A_{\mathrm{MCV}}(5)$.
- For $k \geq 4$, compare the lower bound for $A_{\mathrm{AC}}(n)$ given by inequality (5) with the upper bound for $A_{\mathrm{MCV}}(2n-1)$

$$A_{\mathrm{MCV}}(2n-1) < 1 - \frac{\binom{2n-1}{n}\rho^n}{(1+\rho)^{2n-1}}.$$

A sufficient condition for $A_{\mathrm{AC}}(n) > A_{\mathrm{MCV}}(2n-1)$ is then given by

$$\frac{\binom{2n-1}{n}}{n} \quad > \quad (1+\rho)^{n-1}. \tag{9}$$

This inequality holds for $n = 4$ and any $\rho \le 1$ as $\binom{7}{4}/4 > 8$.

It also holds for all $n > 4$ and any $\rho \le 1$ since

$$\frac{\binom{2n+1}{n+1}}{n+1} = \frac{2n+1}{n+1}\frac{2n}{n+1}\frac{\binom{2n-1}{n}}{n} > 2\frac{\binom{2n-1}{n}}{n}$$

for all $n > 1$. Inequality (10) holds by recurrence for all $n > 4$ and any $\rho \le 1$. □

**Theorem 3.2.** The availability $A_{\mathrm{AC}}(n)$ of a replicated data object with $n$ identical replicas managed by the original available copy consistency protocol is greater than the availability $A_{\mathrm{GQC}}(n)$ of a replicated data object with $2n$ identical replicas managed by the general quorum consensus protocol as long as the failure rate to repair rate ratio $\rho$ remains much less than one.

**Proof**. Consider a replicated data object $X$ with $2n$ replicas managed by the general quorum consensus protocol. Let $o_1, \dots, o_m$ be the $m$ operations defined on $X$ and let $f_i$ be the relative frequency of operation $o_i$. Since different quorums are associated with different operations, the replicated data object has a different availability $A^i_{\mathrm{GQC}}(2n)$ for each operation $o_i$. We can define the *average availability* $A_{\mathrm{GQC}}(2n)$ of $X$ as the average availability of all operations defined on $X$ weighted by their relative frequencies $f_i$:

$$A_{\mathrm{GQC}}(2n) = \sum_{i=1}^{m} f_i A^i_{\mathrm{GQC}}(2n).$$

Assume now that the largest quorum of the most frequent operation $o_j$ is some integer $k \le n$. There must be at least one operation $o_l$ such that one of its quorums intersects with the largest quorum of $o_j$. If $f_1$ is the relative frequency of $o_l$, an upper bound for the average availability of $X$ is then given by

$$A_{\mathrm{GQC}}(2n) = 1 - f_l \frac{\binom{2n}{k}\rho^k}{(1+\rho)^{2n}}.$$

A sufficient condition for $A_{\mathrm{AC}}(n) > A_{\mathrm{GQC}}(2)$ is then given by

$$\frac{n\rho^n}{(1+\rho)^n} < f_l \frac{\binom{2n}{k}\rho^k}{(1+|rho)^{2n}}.$$

This inequality holds for all $f_l$ such that

$$f_l > \frac{n\rho^{n-k}(1+\rho)^n}{\binom{2n}{k}},$$

a condition which will be almost always verified in practice as failure-rate-to-repair rate ratios are typically well below 0.1. For instance, when $n = 3$ and $k = 1$, AC outperforms GQC with twice the number of replicas as long as

$$f_l > \frac{1}{2}\rho^{n-1}(1+\rho)^n,$$

which reduces to $f_l > 0.0067$ for $\rho = 0.1$. □

**Theorem 3.3.** The availability $A_{AC}(n)$ of a replicated data object with $n$ identical replicas managed by the available copy consistency protocol is greater than the availability $A_{DV}(n)$ of a replicated data object with the same $n$ replicas managed by the dynamic voting protocol or the hybrid dynamic voting protocol as long as the failure rate to repair rate ratio $\rho$ remains less than or equal to one.

**Proof**. Dynamic voting and hybrid dynamic voting protocols require at least two current replicas of the replicated data object to be available in order to allow access to the replicated data object. For both protocols

$$A_{\mathrm{DV}}(n) < 1 - \frac{\binom{n}{n-1}\rho^{n-1}}{(1+\rho)^n} - \frac{\binom{n}{n}\rho^n}{(1+\rho)^n} = \frac{n\rho^{n-1}+\rho^n}{(1+\rho)^n}.$$

Since

$$A_{\mathrm{AC}} > 1 - \frac{n\rho^n}{(1+\rho)^n}.$$

$A_{\mathrm{AC}}(n) > A_{\mathrm{DV}}(n)$ for all $\rho \leq 1$.

Note that for all $\rho \leq 1/n$, one can show that $A_{\mathrm{AC}}(n-1) > A_{\mathrm{DV}}(n)$. *qed*

**Theorem 3.4.** The reliability $R_{\mathrm{AC}}(n,t)$ of a replicated data object with $n$ identical replicas managed by the available copy consistency protocol is greater than the reliability $R_{\mathrm{MCV}}(n,t)$ of a replicated data object with $2n-1$ identical replicas managed by the majority consensus voting protocol.

**Proof**. A *1-out-of-n* replicated system is more reliable than an *n-out-of-*$(2n-1)$ system. □

**Theorem 3.5.** The reliability $R_{\mathrm{AC}}(n,t)$ of a replicated data object with n identical replicas managed by the available copy consistency protocol is greater than the reliability $R_{\mathrm{DV}}(n,t)$ of a replicated data object with $n+1$ identical replicas managed by the dynamic voting protocol or the hybrid dynamic voting protocol.

**Proof**. A *1-out-of-n* replicated system is more reliable than a *2-out-of-*$(n+1)$ system. □

## 3.6 Discussion

Two questions still remain unanswered. Available copy protocols have been shown to provide the highest possible reliability figures for all consistency protocols that do not generate new replicas to replace those that have failed. One may wonder how far from optimum are the availabilities provided by these protocols, and ask how well optimistic available copy, naïve available copy and the original available copy protocols fare when compared to the voting protocols.

In the absence of any optimal consistency protocol for replicated data objects, the best alternative is to select as a benchmark the availability of a replicated data object that can be accessed as long as one replica can be accessed. Such 'ideal' protocol does nothing to insure data consistency; it simply provides an upper-bound for the availabilities that could be reached by any consistency protocol that does not regenerate failed replicas.

Figures 8, 9, and 10 display the availabilities achieved by available copy and naïve copy protocols with two, three and four replicas respectively. These availabilities are compared with those provided by majority consensus voting for twice the number of replicas and the upper-bound obtained by not enforcing data consistency. Availabilities achieved by optimistic available copy protocols are not included as they have been shown to fall between available copy and naïve available copy. In all three graphs $\rho$ varies between 0 and 0.2. Zero corresponds to perfectly reliable replicas and 0.2 to replicas that are repaired five times faster than they fail and have an individual availability of 5/6.

Reliabilities for the replicated data objects are displayed in Figure 11 and 12. Each graph compares the reliability function of available copy protocols for 2 and 3 replicas and $\rho = 0.1$ with the reliability function of majority consensus voting with twice the number of replicas. Although reliability functions for $\rho = 0.05$ and 0.2 were computed, they were not included as they did not differ significantly.

Figures 8 to 12 require a few comments. They clearly indicate that all available copy protocols provide much higher availabilities and reliabilities than majority consensus voting. They also show that the availabilities provided by available copy protocols differ only by a narrow margin from the availability provided by an ideal protocol. Hence, it is unlikely that a protocol improving upon the performance of available copy protocols without regenerating failed replicas will be found.

These conclusions need to be qualified as they rely on the hypotheses introduced by our Markovian analysis. It was assumed that network partitions and other partial communication failures were impossible. This assumption holds as long as all replicas of the data object are stored on the same CSMA/CD segment or on the same token ring. But, it precludes the use of available copy protocols in many environments where site holding replicas are separated by gateways, unless the protocol is augmented to detect and reconcile inconsistencies introduced while the network was partitioned [8]. The voting protocols are not subject to this limitation.

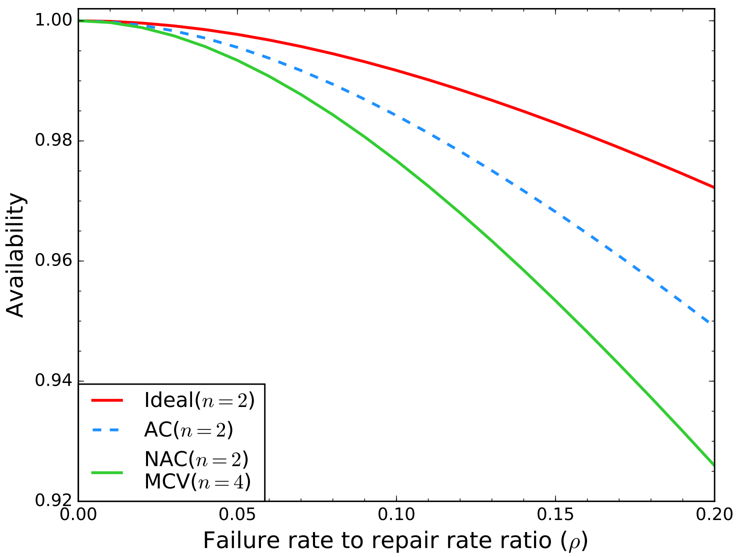

Figure 8: Availabilities for two available copies and four voting copies.

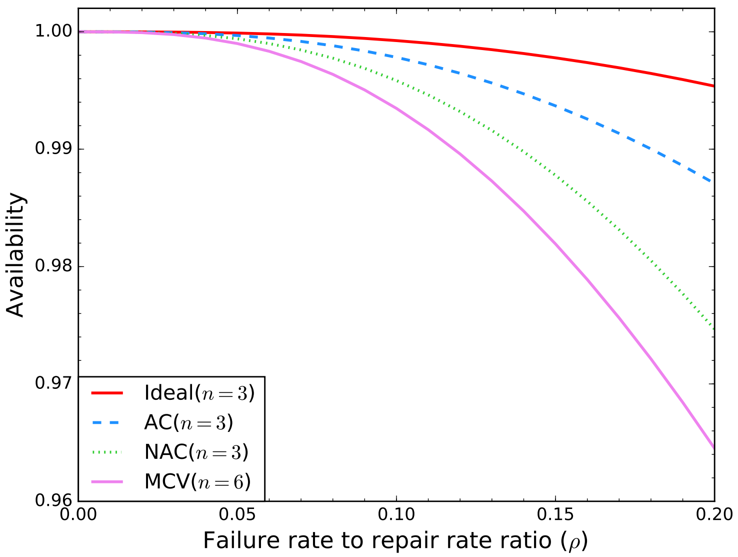

Figure 9: Availabilities for three available copies and six voting copies.

Available copy protocols do not guarantee serializability of concurrent accesses as do voting protocols. To allow concurrent accesses, available copy protocols must be supplemented by a locking protocol if the network does not provide atomic broadcast [4].

Finally, simultaneous failures of all sites holding replicas were not considered. Such failures often result from external events such as power failure or high-voltage transients. Available copy protocols require the recovery of all replicas that were assumed to be available before the failure. This is not a problem as long as none of the sites holding replicas are permanently damaged. Modification of optimistic available copy to disallow accesses when the new was-available set does not contain at least a fixed fraction $q$ of the sites included in the previous was-available set will reduce the risk. The parameter $q$ should be chosen to be well below 0.5 so as not to affect data availability. The modified protocol allows recoveries after a total failure once $\lfloor (1-q)m \rfloor + 1$ of the $m$ sites included

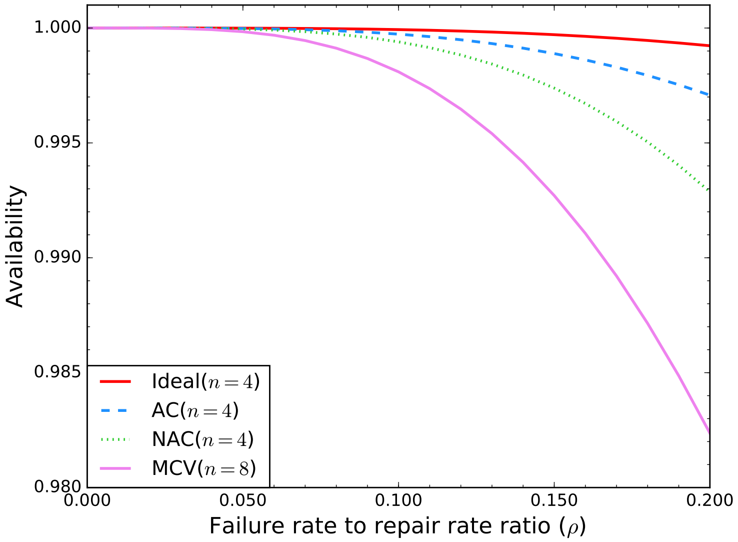

Figure 10: Availabilities for four available copies and eight voting copies.

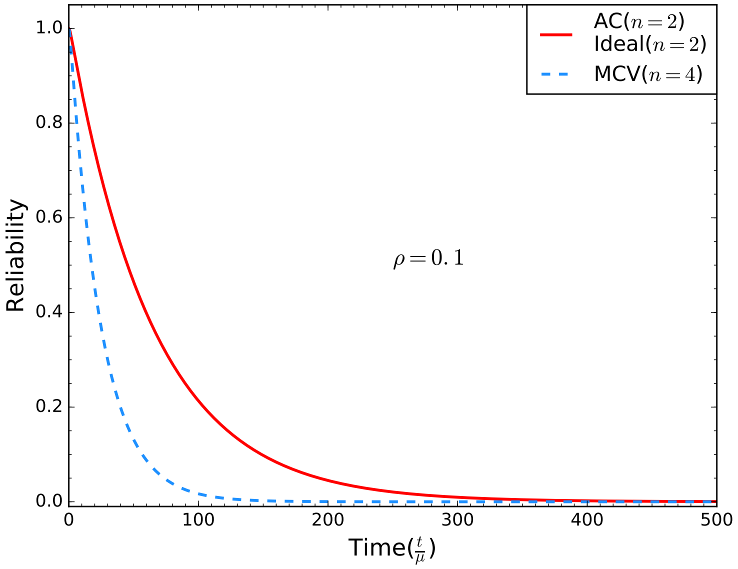

Figure 11: Reliability of two available copies and four voting copies ($\rho$ is the failure rate to repair the rate ratio).

in the most recent was-available set have recovered.

# 4 Traffic analysis

An important, often neglected, aspect of the performance of consistency protocols is the cost in message traffic. In this section, we evaluate the traffic costs of naïve available copy and optimistic available copy protocols and compare them to the cost of majority consensus voting. We do not consider the original available copy protocol as it requires a constant exchange of messages among the available sites and has a traffic cost bounded only by the rate at which the polling messages are generated.

Our analysis focuses on the number of high-level transmissions that occur, such as requests for version num-

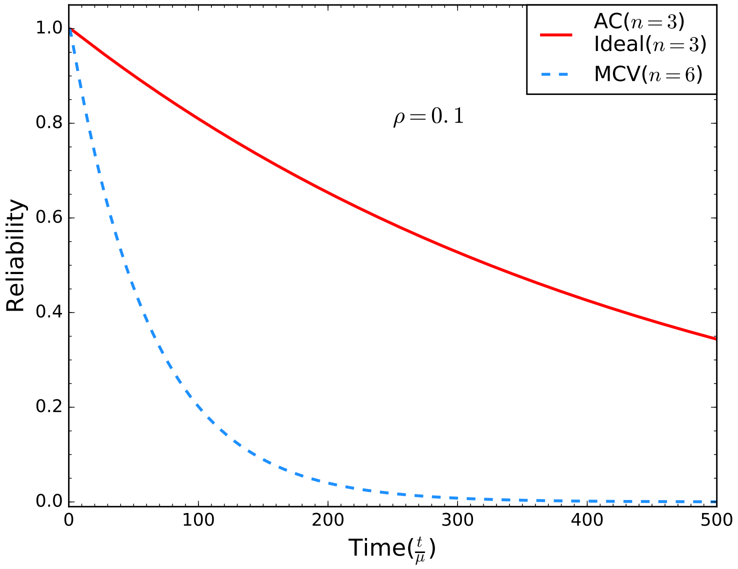

Figure 12: Reliability of three available copies and six voting copies ($\rho$ is the failure rate to repair rate ratio).

bers, block transfers, and the like. The details of the network implementation will determine the actual number of messages generated by a high-level request. We also assume that each write operation requires a two-phase commit protocol to ensure serializability and atomicity. Our results are therefore different from those presented in [6] since that study did not consider the additional message traffic resulting from commit protocols.

We will consider two addressing mechanisms: multicast mechanisms in which a single transmission may be received by several sites, and networks which require transmissions to be addressed to an individual site. The three protocols retain their relative advantages in either type of network, though the differences are amplified in a single destination network.

We make some simplifying assumptions about the model. We assume that in each case the local site holds a replica of the data, a favorable assumption for all protocols since it allows local data access. We consider the case where all sites are operational, which is the most common situation. If this assumption were not made, the voting protocol would suffer since more messages would have to be sent in order to collect a quorum of replicas when failed sites are encountered.

To implement a two-phase commit, naïve available copy protocols and optimistic available copy protocols require three exchanges of information: the coordinator needs to send a *request commit* message to the $n-1$ non-local replicas, these $n-1$ replicas need to return their answers and the coordinator needs then to send them a final *commit* message. The total number of high-level messages exchanged during a write operation is $n_{\mathrm{AC}}^{\mathrm{W}} = 1+(n-1)+1 = n+1$ in a multicast network, and $n_{\mathrm{AC}}^{\mathrm{W}} = 3(n-1)$ in a unicast network. Since the local site holds a replica of the data, all reads can be performed locally and do not result in any network traffic. Hence $n_{\mathrm{AC}}^{\mathrm{R}} = 0$.

Under majority consensus voting, dynamic voting, linear-dynamic voting and hybrid dynamic voting, read and write operations need to consult a majority of replicas. Write operations will therefore require $n_{\mathrm{V}}^{\mathrm{W}} = \lfloor n/2 \rfloor + 2$ high-level messages in a multicast network, and $n_{\mathrm{V}}^{\mathrm{W}} = 3\lfloor n/2 \rfloor$ messages in a unicast network. We should point out that these figures represent strict minima: very few voting protocols implement writes by updating exactly $\lfloor n/2 \rfloor + 1$ replicas of the data object since a failure of any of these $\lfloor n/2 \rfloor + 1$ replicas would make the data object temporarily unavailable.

As the replicas participating in a read do not need to commit, read operations only require $n_{\mathrm{V}}^{\mathrm{R}} = 1 + \lfloor n/2 \rfloor$ high-level messages in a multicast network, and $n_{\mathrm{V}}^{\mathrm{R}} = 2\lfloor n/2 \rfloor$ messages in a unicast network.

Read operations can be made less expensive by using *quorum consensus voting* and reducing the read quorum. This would however result in a larger write quorum, which would increase the cost of read operations and lower the write availability of the data object.

If $r$ is the read to write ratio, the average number of high-level messages exchanged by the available copy protocol during an operation is

$$n_{\mathrm{AC}} = \frac{n+1}{r+1}$$

in a multicast network, and

$$n_{\mathrm{AC}} = \frac{3(n-1)}{1+r}$$

in a unicast network while voting requires an average of

$$n_{\mathrm{V}} = \lfloor n/2 \rfloor + \frac{2+r}{1+r}$$

messages in a multicast network, and an average of

$$n_{\mathrm{V}} = \frac{3+2r}{1+r} \lfloor n/2 \rfloor$$

messages in a unicast network.

Available copy protocols have a lower message overhead than voting protocols as long as

$$r > \frac{\lceil n/2 \rceil - 1}{\lfloor n/2 \rfloor + 1}$$

in a multicast network or

$$r > \frac{3(\lceil n/2 \rceil - 1)}{2\lfloor n/2 \rfloor}$$

in a unicast network. These conditions are met for most data objects as research on observed access patterns have shown the read to write ratios of typical computer systems to be in the neighborhood of 2.5 : 1 [22].

The comparison is even more favorable to available copy protocols in situations where a single writer policy can be enforced. Since available copy protocols only require commits to ensure serializability of writes, the number of messages exchanged during a write operation becomes $n'^{\mathrm{W}}_{\mathrm{AC}} = 1 + (n-1) = n$ in a multicast network, and $n'^{\mathrm{W}}_{\mathrm{AC}} = 2(n-1)$ in a unicast network. Besides, a single response from any replica is sufficient to guarantee that a write is successful. This property has been used in the *Gemini* replicated file system to improve read and write access times [5]. *Gemni* enforces a single writer policy and uses a *semi-synchronous* write policy that combines a voting protocol at open time and an available copy approach during file access. As a result reads can be performed on the closest current replica of the file and writes can return after having received the response of a single replica.

However, there are some data objects for which available copy protocols are not optimal. These objects, such as logs and mailboxes, typically are more often updated than they are read. General quorum consensus would result in a smaller message overhead as it allows inexpensive writes at the cost of more expensive reads.

# 5 Conclusions

Available copy protocols have not yet received the attention they deserve because they were believed to be hard to implement and their performance never fully understood. We have investigated two available copy protocols that are easy to implement and provide superior availabilities and reliabilities. The first protocol, *naïve available copy*, does not maintain state information and waits for the recovery of all sites holding replicas following a total failure. The second protocol, *optimistic available copy*, maintains state information at write and recovery time and performs nearly as well as protocols assuming instantaneous detection and propagation of this information.

Markov models were used to compare the performance of these protocols with those afforded by the original available copy protocol, majority consensus voting, general quorum consensus and an ideal protocol allowing unrestricted access. We found that available copy protocols provided the highest possible reliability for any consistency protocol that does not generate new replicas to replace those that failed. We also found that available copy protocols yielded availabilities much superior to those obtained with majority consensus voting or general quorum consensus with twice the number of replicas.

Naïve available copy and optimistic available copy were found to yield a lower message traffic than voting protocols in the range of read-to-write normally found in most installations.

Three major conclusions can be drawn from this study. First, available copy protocols can be implemented efficiently and constitute the method of choice to manage replicated data objects in environments where partitions are excluded. This will be the case when all replicas are on the same CSMA/CD segment, the same token ring, or when the protocol is augmented to detect and reconcile inconsistencies introduced while the system was partitioned. Second, there is very little difference between the availabilities provided by naïve available copy and optimistic available copy protocols for the small values of the failure rate to repair rate ratio typical of most modern hardware. Finally, it is highly unlikely that any consistency protocol improves on the performance of available copy protocols unless it regenerates failed replicas.

Further research in the area should focus on protocols that are resilient to partial communication failures. A promising avenue of research is the development of voting protocols that take into account the topology of the network, like *topological dynamic voting* [26] or *voting with ghosts* [34]. Preliminary simulation results indicate that these protocols effectively bridge the performance gap that exists between available copy protocols and voting protocols.

## Acknowledgement

We wish to thank Walter Burkhard, Keith Messer, Bruce Martin, Alexander Glockner and all the members of the Gemini group for their help and encouragement. We are also grateful to John Carroll for his many contributions to this work. The second author is especially indebted to Ernestine McKinney for her assistance and encouragement.

This work has been done with the aid of MACSYMA,a large symbolic manipulation program developed at the Massachusetts Institute of Technology. MACSYMA a trademark of Symbolics, Inc.

## References

[1] D. Barbará, H. Garcia-Molina, and A. Spauster. Policies for Dynamic Vote Reassignment. In *Proceedings of the Sixth International Conference on Distributed Computing Systems (ICDCS '86)*, pp. 37–44, 1986.

[2] P. A. Bernstein and N. Goodman. An Algorithm for Concurrency Control and Recovery in Replicated Distributed Databases. *ACM Transactions on Database Systems (TODS)*, 9(4):596–615, 1984.

[3] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley Publishing Company, 1987.

[4] K. P. Birman. Replication and fault-tolerance in the isis system. In *Proceedings of the 10th ACM Symposium on Operating Systems*, 1985.

[5] W. A. Burkhard, B. E. Martin, and J.-F. Pâris. The Gemini replicated file system test-bed. In *Proceedings of the Third IEEE International Conference on Data Engineering*, pp. 441–448. IEEE, 1987.

[6] J. L. Carroll, D. D. E. Long, and J.-F. Pâris. Block-Level Consistency of Replicated Files. In *Proceedings of the Seventh International Conference on Distributed Computing Systems (ICDCS '87)*, pp. 146–153, 1987.

[7] D. Davcev and W. A. Burkhard. Consistency and Recovery Control for Replicated Files. In *Proceedings of the Tenth ACM Symposium on Operating Systems Principles (SOSP '85)*, pp. 87–96, 1985.

[8] S. B. Davidson, H. Garcia-Molina, and D. Skeen. Consistency in Partitioned Networks. *Computing Surveys*, 17(3):341–370, Sept. 1985.

[9] D. P. Gaver. Stochastic modeling: Ideas and techniques. In G. Louchard and G. Latouche, editors, *Probability Theory and Computer Science*. Academic Press, London, 1983.

[10] D. K. Gifford. Weighted Voting for Replicated Data. In *Proceedings of the Seventh ACM Symposium on Operating Systems Principles (SOSP)*, pp. 150–162. ACM, 1979.

[11] N. Goodman, D. Skeen, A. Chan, U. Dayal, S. Fox, and D. Ries. A Recovery Algorithm for a Distributed Database System. In *Proceedings of the Second ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pp. 8–15. ACM, 1983.

[12] M. Hammer and D. Shipman. Reliability mechanisms for sdd-1: a system for distributed databases. *ACM Transactions on Database Systems (TODS)*, 5(4):431–466, 1980.

[13] M. Herlihy. A quorum-consensus replication method for abstract data types. *ACM Transactions on Computer Systems (TOCS)*, 4(1):32–53, 1986.

[14] S. Jajodia. Managing replicated files in partitioned distributed database systems. In *Proceedings of the Seventh International Conference on Distributed Computing Systems (ICDCS)*, pp. 412–418. IEEE, 1987.

[15] S. Jajodia and D. Mutchler. Integrating Static and Dynamic Protocols to Enhance File Availability. In *Proceedings of the Fourth International Conference on Data Engineering*, pp. 144–154. IEEE, 1988.

[16] D. D. E. Long. *The Management of Replication in a Distributed System.* PhD thesis, Department of Computer Science and Engineering, University of California, San Diego, 1988.

[17] D. D. E. Long and J.-F. Pâris. On Improving the Availability of Replicated Files. In *Proceedings of the Sixth Symposium on Reliable Distributed Systems (SRDS '87)*, pp. 77–83, Williamsburg, Mar. 1987. IEEE.

[18] D. D. E. Long and J.-F. Pâris. A Realistic Evaluation of Optimistic Dynamic Voting. In *Proceedings of the Seventh Symposium on Reliable Distributed Systems (SRDS '88)*, pp. 129–137, Columbus, Oct. 1988. IEEE.

[19] D. D. E. Long and J.-F. Pâris. Regeneration Protocols for Replicated Objects. In *Proceedings of the Fifth International Conference on Data Engineering (ICDE '89)*, pp. 538–545, Los Angeles, Feb. 1989. IEEE.

[20] M. A. McGregor. Approximation formulas for reliability with repair. *IEEE Transactions on Reliability*, R-12:64–92, 1963.

[21] J. Noe and A. Andreassian. Effectiveness of Replication in Distributed Computing Networks. In *Proceedings of the Seventh International Conference on Distributed Computing Systems*, pp. 508–513, 1987.

[22] J. K. Ousterhout, H. D. Costa, D. Harrison, J. A. Kunze, M. Kupfer, and J. G. Thompson. A Trace-Driven Analysis of the UNIX 4.2 BSD File System. In *Proceedings of the 10th ACM Symposium on Operating Systems Principles*, pp. 15–24, Dec. 1985.

[23] J.-F. Pâris. Voting with a Variable Number of Copies. In *Proceedings of the 16th International Symposium on Fault-Tolerant Computing (FTCS '86)*, pp. 50–55, 1986.

[24] J.-F. Pâris. Voting with Witnesses: A Constistency Scheme for Replicated Files. In *Proceedings of the Sixth International Conference on Distributed Computing Systems (ICDCS '86)*, pp. 606–612, 1986.

[25] J. F. Pâris and W. A. Burkhard. On the Availability of Dynamic Voting Schemes. *University of California San Diego Department of Computer Science and Engineering Computer Science*, 1986. Technical Report 86-090.

[26] J.-F. Pâris and D. D. E. Long. Efficient Dynamic Voting Algorithms. In *Proceedings of the Fourth International Conference on Data Engineering*, pp. 268–275. IEEE, 1988.

[27] J.-F. Pâris, D. D. E. Long, and A. Glockner. A Realistic Evaluation of Consistency Algorithms for Replicated Files. In *Proceedings of the 21th Annual Simulation Symposium (SS '88)*, pp. 121–130. IEEE Computer Society Press, 1988.

[28] G. Popek, B. Walker, J. Chow, D. Edwards, C. Kline, G. Rudisin, and G. Thiel. Locus a network transparent, high reliability distributed system. In *Proceedings of the ACM SIGOPS Operating Systems Review*, volume 15, pp. 169–177. ACM, 1981.

[29] C. Pu. *Replication and Nested Transactions in the Eden Distributed System.* Ph.D. dissertation, University of Washington, 1986.

[30] R. D. Schlichting and F. B. Schneider. Fail Stop Processors: An Approach to Designing Fault-Tolerant Computing Systems. *ACM Transactions on Computer Systems (TOCS)*, 1(3):222–238, 1983.

[31] D. Skeen. Determining the Last Process to Fail. *ACM Transactions on Computer Systems (TOCS)*, 3(1):15–30, 1985.

[32] M. D. Skeen. *Crash recovery in a distributed database system.* University of California, Berkeley, 1982.

[33] R. H. Thomas. A Majority Consensus Approach to Concurrency Control. *ACM Transactions on Database Systems (TODS)*, 4(2):180–209, 1979.

[34] R. Van Renesse and A. S. Tanenbaum. Voting with Ghosts. In *Proceedings of the Eighth International Conference on Distributed Computing Systems*, pp. 456–462. IEEE, 1988.