A Realistic Evaluation of Optimistic Dynamic Voting

Darrell D. E. Long
Jehan-François Pâris

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

Proceedings of the Seventh Symposium on Reliable Distributed Systems, Columbus, OH, 1988, pp. 129–137.

Abstract

When data is replicated, an access protocol must be chosen to insure the presentation of a consistent view of the data. Protocols based on quorum consensus provide good availability with the added benefit of mutual exclusion. Of the protocols based on quorum consensus, the *dynamic voting* protocols provide the highest known availability.

We describe a dynamic voting protocol that does not need the instantaneous state information required by the same performance as the original dynamic voting in the asymptotic case, and quickly converges to it for realistic access rates. Our protocol does this at a cost in network similar to that of statistic majority consensus voting.

The first realistic analysis of the availability afforded by dynamic voting protocols is presented, taking the access frequency into account. The analysis confirms our hypothesis that delaying state information does not appreciably affect availability. Discrete event simulation is used to confirm and to extend the results we obtain using analytic models.

Keywords– file consistency, fault-tolerant systems, replicated files, majority consensus voting.

1 Introduction

In a distributed system, data are often replicated for protection against site failures and network partitions. Through the use of replication, increased availability and reliability can be obtained. The *availability* of a data object is the probability that the system is in a state where an access request for that data object will be granted. The *reliability* of a data object is the probability that a time t there has been no sequence of failures such that access requests would be denied at any time during the interval beginning at the initial configuration.

When data is replicated at several sites an access policy must be chosen to insure a consistent view of the data so that it appears as though there were only a single replica of the data. The view presented to the user must remain consistent even in the presence of site failures and network partitions. Sites recovering from a failure must present the data stored at that site in such a way that it is consistent with the global view of the data.

Consensus bases protocols provide consistency in the presence of site failures and network partitions. They also provide the added benefit of mutual partitions. The simplest consensus protocol is *static majority consensus voting* [1, 2, 3, 4, 5]. Static majority consensus voting provides consistency control and mutual exclusion, but does not provide the highest possible availability since it requires that a majority of the sites be accessible for an access request to be granted.

Authors' present addresses: D.Long, CIS Board of studies, University of California, Santa Cruz, Ca 956064; J,-F Pâris, Department of Computer Science, University of Houston, Houston, TX 77004. This work was supported in part by a grant from the NCR Corporation and the University of California MICRO program

An attempt to remedy the short-comings of static majority consensus voting, known as *Dynamic Voting*, was introduced by Davcev and Burkhard [6]. Their protocol improves the performance by allowing quorums to be adjusted automatically during system operation. It is an idealized protocol based on instantaneous state information. Such information cannot be maintained, nor can it even be realistically approximated because of the load it imposes on the system. In this paper, we describe a protocol that provides the same performance as dynamic voting in the asymptotic case, at a cost in message traffic similar to static majority consensus voting.

Our protocol, called *Optimistic Dynamic Voting*, operates on possible out-to-date information, hoping for the best. The protocol provides mutual exclusion and data consistency is preserved. Our protocol improves upon similar work by Jajodia and Mutchler, [7, 8] on dynamic voting and Jajodia's work on linear ordering [9]. There are many benefits to our protocol, including efficiency and ease of implementation. We believe that our protocol is superior due to its flexibility allowing it to easily accommodate linear orderings [9], topological information [10], witness copies [11] and regeneration strategies.

This paper is organized into six sections. Section two describes earlier work in the area of consensus protocols; Section three describes our protocol in detail, discussing the various algorithms involved; Section four contains a stochastic analysis of the availability of the data afforded by our protocol; Section five presents our simulation results. Our conclusions appear in section six.

2 Previous Work

Consensus protocols insure the consistency of replicated data by honoring tread and write requests only when an appropriate quorum of the sites holding replicas of the data can be accessed. In their simplest form, consensus protocols assume that the correct state of replicated data is the state of the majority of the replicas. Ascertaining the state of replicated data requires collecting a quorum of the replicas. Should this be prevented by one or more site failures, the data is considered to be unavailable. We call this protocol *static majority consensus voting*.

This protocol can be refined by introducing different quorums for read and write operations or by allocating different weights, including none, to each replica [3]. Consistency is guaranteed so long as the write quorum is high enough to disallow simultaneous writes on two disjoint subsets of the replicas, and the read quorum is high enough to disallow simultaneous reads and writes on two disjoint subsets of the replicas. A simple extension to static majority consensus voting is the introduction of *witness* copies which do not hold a replica of the data but can attest to its state [11]

The weakness of the static protocols is that the quorum is fixed; it cannot change once the system has begun operation. Because of this, a few failures can render the data inaccessible. Davcev and Burkhard [6] proposed a solution to this problem, known as *Dynamic Voting*. Their policy adjusts the necessary quorum of physical replicas required for an access operation without manual intervention. Quorum adjustments are accomplished by modifying the set of replicas that are allowed to participate in the election. This set is called the *majority partition*. To ensure consistency, any new majority partition is required to contain a majority of the replicas in the previous majority partition.

The basis of dynamic voting is the *connection vector*. The connection vector instantaneously records the state of the network with respect to each site. Each physical replica of the replicated data object has an associated ensemble of state information consisting of a version number and a partition vector. The *version number* of a physical replica represents the number of successful write operations to the physical replica. The *partition vector* at a site records the version number of all sites with respect to the site. In its original form, dynamic voting allows accesses to proceed so long as a strict majority of the *current* physical replicas are accessible.

In situations where the number of current physical replicas within a group of mutually communicating sites is equal to the number of current replicas not in communication, dynamic voting cannot proceed and declares the replicated file to be inaccessible. An extension proposed by Jajodia [9], known as *Dynamic-linear* enhances dynamic voting by resolving ties by applying a total ordering to the sites. This simple extension greatly enhances the availability.

This instantaneous dynamic-linear protocol accommodates the situation by using an arbitrary, but fixed, tie breaking rule. The sites holding replicas of the data are given a static linear ordering. Then, when a tie occurs, if the group of communicating sites contains exactly one-half the current physical replicas and that group contains the maximum element among the group of current physical replicas, then that group is declared to be the majority partition.

Our experiments have shown that the connection vector is an impossible object to implement. Attempts to approximate it consume nearly all the available machine cycles on a moderate sized site [12]. Because dynamic voting provides very high availability, but is not practical in its original form, we began to look for alternative implementations.

We had previously developed an efficient protocol implementing the available copy protocol using *was-available sets* [13, 14]. The was-available sets were stored at each site and contained the identity of the sites that participated in the last operation. By maintaining the was-available sets only at access time, we were able to implement the available copy protocol at a low cost. We decided to extend this approach to develop a similar solution for dynamic voting. Section three discusses our dynamic voting protocol, which we first presented in [10].

There has been recent work in the area of dynamic voting protocols by Jajodia and Mutchler [7, 8]. They have developed a protocol based on the number of sites that participate in a write operation. Their original protocol did not accommodate lexicographic ordering [7], but was added in a later paper [8] by requiring more state information to be kept.

Another approach to dynamic consensus protocols was pioneered by Garcia-Molina et al. [15, 16, 17, 18]. Their protocols are complementary to ours in that they operate by assigning increasing numbers of votes to the surviving sites following a site failure. These protocols fall into two categories: synchronous policies, known as *group consensus*, and asynchronous policies. The asynchronous policies again fall into two categories: *alliance*, where the surviving sites join together to supplants the failed sites, and *overthrow*, where a single site supplants the failed site. When a site recovers following a failure it must rejoin the majority partition either by playing *catch-up* and increasing its votes, or by causing the other sites to decrease the number of votes that they hold.

3 Optimistic Consensus Protocols

We have found that correctness does not depend on instantaneous state information and that dynamic voting protocols exists that operate correctly using possibly out-of-date information. Using information that is out-of-date does not affect the consistency of the data, but does sacrifice some availability. Since our protocol propagates system state information when an access is successfully made, the amount of availability that is lost is related to the rate at which the data is accessed.

There are three sets of information that must be maintained at each site: the partition set, P_i, which represents the set of sites which participated in the last successful operation, an operation number, o_i, and a version number, v_i. This information is stored at each site and is modified when an access occurs.

The operation numbers are introduced to speed the recovery of a site, supplementing the information provided by the version numbers. There is a trade-off between the extra maintenance of the operation number and increased recovery time. If the version number is incremented on each read operation, then recovery will be forced to occur when it is unnecessary. If version numbers alone were used, and were not incremented for each read operation, then there would be cases where multiple majority partitions could exist. This is because a read operation does not change the state of the data, and so it should not change the version number of the data. Instead, we chose to implement the partition set as a replicated data object with loose consistency constraint.

The basis of our method is the protocol for detecting whether the access request originates within the majority partition. Since there can be only one majority partition, mutual exclusion is guaranteed and consistency is preserved.

Protocol 1. Protocol for deciding whether the current partition is the majority partition

1. Find the set of all sites communicating with the requesting site, call it $\mathbf{R}$.

2. Requests from each site $i \in \mathbf{R}$ its partition set P_i, its operation number o_i, and its version number v_i.

3. Let $\mathbf{Q} \subset \mathbf{R}$ be the set of all sites with operation numbers that match that of the site with the highest operation number.

4. Let P_m be the partition set of any site in **Q**.
5. If the cardinality of **Q** is greater than one half the cardinality of P_m, or is the exactly one half and contains the maximum element of P_m then the current partition is the majority partition.

The algorithm for performing a read operation is simple. It first ascertains whether the current partition is the majority partition. This is done in the same way for all of the algorithms presented. A message is broadcast to all sites requesting their partition set, operation number and version number; those that send replies are considered to be in a current partition. The set of current sites is found by computing the maximum operation number of all of the sites. It is this set of sites that will participate in the operation. The set of sites holding up-to-date replicas is called the *quorum set*. If the quorum set represents a majority of the previous quorum, represented by P_i, then the access request is granted. If there is a tie, that is exactly one half of the previous quorum, then a total ordering on the set of sites is used to decide if access will be granted.

Once it has been ascertained that the current partition is the majority partition, then access can continue. The read operation is performed and the operation number is incremented and sent along with the set of current sites to each of the current sites to serve as their new partition sets. This last action serves to modify the quorum required for access requests to be granted in the future.

```
procedure READ
begin
    let $U$ be the set of all sites participating in the replication
    $\langle R, \mathbf{o}, \mathbf{v}, \mathbf{P} \rangle \leftarrow \mathrm{START}(U)$
    $Q \leftarrow \{r \in R : o_r = \max_{s \in R}\{o_S\}\}$
    $S \leftarrow \{r \in R : v_r = \max_{s \in R}\{v_S\}\}$
    choose any $m \in Q$
    if $(|Q| > \frac{|P_m|}{2}) \vee (|Q| = \frac{P_m}{2} \wedge \max(P_m) \in Q)$ then
        perform the read
        $\mathrm{COMMIT}(S, o_m+1, v_m, S)$
    else
        $\mathrm{ABORT}(R)$
    fi
end READ
```

Figure 1: Read Algorithm

There are several items in the algorithm for reading that require explanation. The START operation begins the operation and returns R which is the set of reachable sites and three arrays; **o, v** and **P** which are the operation numbers, version numbers, and partition sets respectively. The COMMIT operation completes the operation and transmits the new consistency control information to all up-to-date replicas.

The algorithm for writing is similar to the algorithm for reading. Again, it is ascertained if the current partition is the majority partition. If this is successful, proceed as in the protocol for reading. The write operation is performed. The operation number and the version number are incremented and sent along with the set of current sites to all of the current sites to serve as their new partition sets.

```
procedure WRITE
begin
    let $U$ be the set of all sites participating in the replication
    $\langle R, \mathbf{o}, \mathbf{v}, \mathbf{P} \rangle \leftarrow \text{START}(U)$
    $Q \leftarrow \{r \in R : o_r = \max_{s \in R}\{o_S\}\}$
    $S \leftarrow \{r \in R : v_r = \max_{s \in R}\{v_S\}\}$
    choose any $m \in Q$
    if $(|Q| > \frac{|P_m|}{2}) \vee (|Q| = \frac{P_m}{2} \wedge \max(P_m) \in Q)$ then
        perform the write
        COMMIT($S, o_m+1, v_m+1, S$)
    else
        ABORT($R$)
    fi
end WRITE
```

Figure 2: Write Algorithm

The recovery algorithm begins as do the other algorithms, ascertaining whether the current partition is the majority partition. If the recovering site is able to communicate with the majority partition, then it determines whether the replica at that site is up-to-date. If it is not, then it must be copied from one of the sites in the quorum set. The recovering site then sends the union of the set of current sites and itself to all of the current sites, including itself, to serve as their new partition sets.

The advantage of our protocol is that it is nearly as efficient as static majority consensus in terms of the number of messages sent, and that its implementation is simple. There are no assumptions made about the state of the network other than that of which can be found by examining the partition sets and version numbers. We have an advantage over the protocol proposed by Jajodia in that we can easily incorporate linear ordering, topological information, witness copies and regeneration strategies into the decision process.

We believe that our protocol is superior to the dynamic-linear protocol [7, 8] since it can easily accommodate linear orderings [9], topological information [10], witness copies [11] and regeneration strategies. The partition set can be easily implemented as a bit vector, resulting an efficient implementation. We feel that due to its flexibility, low cost and simple implementation, our policy is the protocol of choice for replicated data consistency and mutual exclusion.

4 Stochastic Analysis

In this section, we present an analysis of availability provided by our protocol. The previous work on estimating the availability of replicated data managed by dynamic voting protocols [7, 8, 19] had assumed idealized consistency control protocols that possessed instantaneous information about the system state. Such policies are unrealistic since instantaneous information is an unachievable goal in a distributed system and in attempting to approximate it a crippling load can be imposed on the sites [12].

The protocols that we present are realistic since they do not rely on information that is impossible to obtain. They are efficient in the sense that the message traffic incurred by them is similar to that incurred by static majority consensus voting. The performance of or protocol is characterized by the rate at which access requests occur. When the access requests are more frequent, the site availability information is closer to the true state of the system and availability improves. This is reflected in the analysis where access rates are explicitly considered.

Our model consists of a set of sites with independent failure modes connected via a network which does not fail. When a site fails, a repair process is immediately initiated at the site. Should several sites fall, the repair process will be performed in parallel on those failed sites. We assume that failures are exponentially distributed with mean λ, that repairs are exponentially distributed with mean μ and that access requests are characterized by

```
procedure RECOVER
begin
    repeat
        let $I$ be the recovering site
        let $U$ be the set of all sites participating in the replication
        $\langle R,\mathbf{o},\mathbf{v},\mathbf{P}\rangle \leftarrow \mathrm{START}(U)$
        $Q \leftarrow \{r \in R : o_r = \max_{s\in R}\{o_S\}\}$
        $S \leftarrow \{r \in R : v_r = \max_{s\in R}\{v_S\}\}$
        choose any $m \in Q$
        if $(|Q| > \frac{|P_m|}{2}) \vee (|Q| = \frac{P_m}{2} \wedge \max(P_m) \in Q)$ then
            if $v_i < v_m$ then
                copy the file from the site $m$
            fi
                $\mathrm{COMMIT}(S \cup \{I\}, o_m+1, v_m, S \cup \{I\})$
        else
            $\mathrm{ABORT}(R)$
        fi
    until successful
end RECOVERY
```

Figure 3: Recovery Algorithm

a Poisson process with mean κ. The system is assumed to exist in statistical equilibrium and to be characterized by a discrete-state Markov process.

The assumptions that we have made are required for a steady-state analysis to be tractable [20]. The assumptions about failure, repair and access distributions are required to preserve the Markov property and thus keep the number of states finite. If the network is allowed to partition, then the number of topologies that must be analyzed is greater than exponential. Thus, this analysis gives an optimistic view of the three protocols under consideration. For this reason, discrete event simulation is used to confirm our analytic results. In section five, we will study some topologies using discrete event simulation to model networks where partitions and distinct or non-exponential failure modes are possible

Definition 4.1 The availability of a replicated data object consisting of n replicas and managed by a consistency protocol **S**, denoted $\mathbf{A}_S(\mathbf{n})$, is the stationary probability of the system being in a state permitting access.

The state transition diagram for three replicas is shown in figure 4. The states represented by unmarked states where an access requested would be granted, and are called *active* states. The states marked with a bar represent states where an access request would be denied, and are called *comatose* states. The states are labeled by tuples. The first coordinate represents the number of sites which are believed to be active, the second coordinate represents the actual number of active site. The transition between states fall into several categories.

1. *Failure transitions.* The failure transition represents the event of a site failure. There is no exchange of site availability information. The state transition $\langle 3,3\rangle \rightarrow \langle 3,2\rangle$, with rate 3λ, is an example of such a transition.

2. *Recovery transitions.* The recovery transition represents the event of sire being activated following a repair. The information concerning site availability is exchanged at this time. The state transition $\langle 2,2\rangle \rightarrow \langle 3,3\rangle$, with the rate μ, is an example of such a transition.

3. *Access transitions.* An access transition occurs when an access request is made and is granted. The information concerning site availability is exchanged at this time. The state transition $\langle 3,2\rangle \rightarrow \langle 2,2\rangle$, with a rate κ, is an example of such a transition

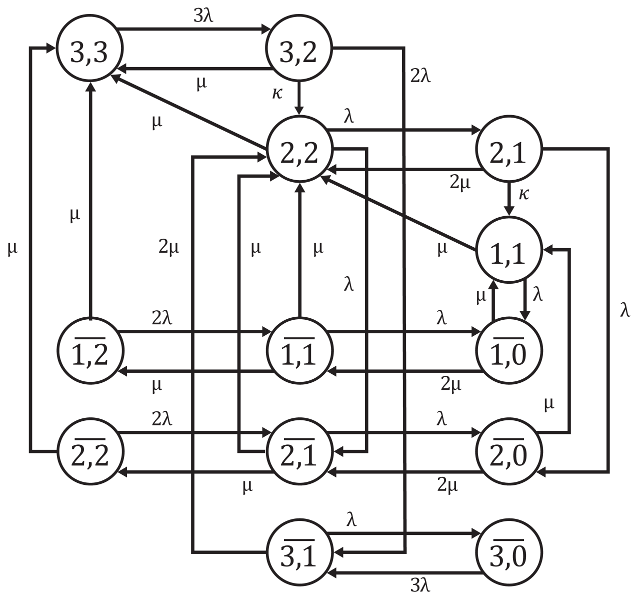

Figure 4: State Transition Diagram for Three Sites

The probability of the system being in an active state $\langle i,j\rangle$ where access is permitted is represented by $p_{i,j}$. The probability of being in a comatose state $\overline{\langle i,j\rangle}$ where access would be denied, is represented by $q_{i,j}$. The state diagram, along with the boundary condition $\sum_{i,j} p_{i,j} + \sum_{i,j} q_{i,j} = 1$, yields a set of equations that is solved using standard techniques. Symbolic manipulation software is essential because, although the process is simple, it is tedious and error-prone.

The availability is given by the sum of the probabilities of being in a state where access is permitted and for three replicas is given by:

$$\begin{aligned} A_O(3) &= \sum_{i,j} p_{i,j} \\ &= \frac{2\rho^4 + \phi\rho^3 + 6\rho^3 + 3\phi\rho^2 + 11\rho^2 + 4\phi\rho + 6\rho + \phi + 1}{(\rho+1)^4(2\rho+\phi+1)} \end{aligned}$$

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

The availability expression for four replicas is not presented due to its size. Due to the complexity of the expressions, a closed-form solution for A_O(n) is difficult to obtain. Neither does it seem fruitful to search for it since the number of replicas will in practice be unlikely to exceed five or six because of the storage costs and because of the availability that these protocols provide.

The availability provided by optimistic dynamic voting is related to the availability provided by dynamic-linear by the rate at which access requests occur. As the access rate increases, the information available to our protocol regarding the system state becomes closer to the true state of the system and the availability increases. So long as the access rate is greater that the failure rate, the performance of our protocol is very good; regardless of the access rate it is always superior to static majority consensus voting.

Theorem 4.1 The availability provided by Optimistic Dynamic Voting, A_O(n), approaches the availability provided by instantaneous Dynamic-linear, A_L(n), as the access rate approaches infinity.

This can be shown by direct manipulation for small numbers of replicas, as it is below for three.

$$\lim_{\phi\to\infty} A_O(3) = \frac{\rho^3 + 3\rho^2 + 4\rho + 1}{(\rho+1)^4} = A_L(3)$$

If we consider a state ⟨i,j⟩, i>j then an access transition will place the system in state ⟨j,j⟩. As the access rate approaches infinity, the time spent in state ⟨i,j⟩ goes to zero. State ⟨j,j⟩ effectively replaces state ⟨i,j⟩, resulting in exactly the state diagram for instantaneous dynamic-linear [19].

We can now compare the availability provided by our protocol with that of other policies. For comparison, we have chosen instantaneous dynamic-linear and static majority consensus voting because they provide an upper bound and a lower bound, respectively, on the availability afforded by our protocol. The availability provided by static majority consensus voting [19]

$$A_V(n) = \sum_{j=n}^{\lceil n/2 \rceil} \frac{\binom{n}{n-j}\rho^{n-j}}{(1+\rho)^n}$$

For an odd number of replicas, and by dynamic-linear for three and four replicas [19]

$$A_L(3) = \frac{\rho^3 + 3\rho^2 + 4\rho + 1}{(\rho+1)^4}$$

$$A_L(4) = \frac{6\rho^6 + 35\rho^5 + 102\rho^4 + 152\rho^3 + 113\rho^2 + 39\rho + 6}{(\rho+1)^4(6\rho^3 + 17\rho^2 + 15\rho + 6)}$$

are given for reference.

The graph in figure 5 shows the compared availabilities for three replicas managed by instantaneous dynamic-linear, static majority consensus voting and optimistic dynamic voting with a representative access rate ϕ. The improvement in availability provided by instantaneous dynamic-linear over static majority consensus voting for three copies is not great. As predicted, the availability of data provided optimistic dynamic voting is related to the access rate and lies between the two other policies.

When four replicas are considered, the improvement in the availability provided by instantaneous dynamic-linear over static majority consensus voting is easily seen. The graph in figure 6 shows the compared availabilities for four replicas managed by the same set of policies, for several access rate ϕ. The performance of optimistic dynamic voting quickly converges to that of instantaneous dynamic-linear. Even for modest access rates, the advantage of using optimistic dynamic voting over using static majority consensus voting can clearly be seen. The availability provided by optimistic dynamic voting is an improvement over static majority consensus voting even when site availability information is exchanged only at recovery time. The availability of the data managed by optimistic dynamic voting rapidly approaches that of dynamic-linear even for low access rates.

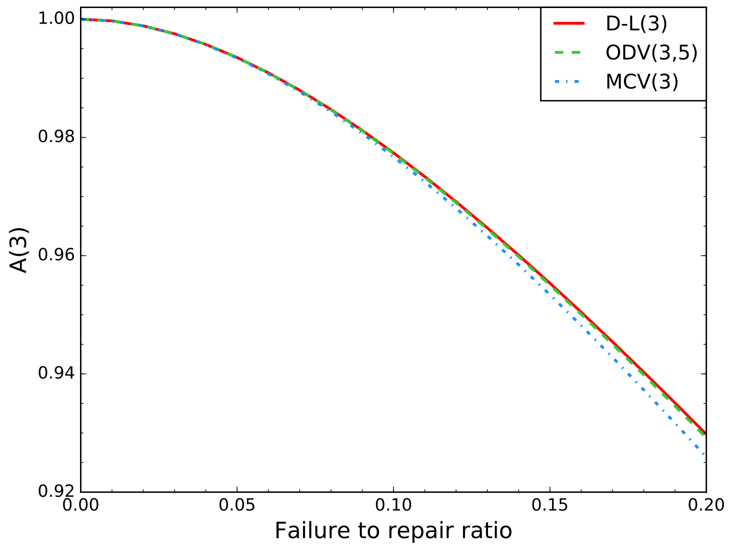

Figure 5: Compared Availabilities for Three Copies

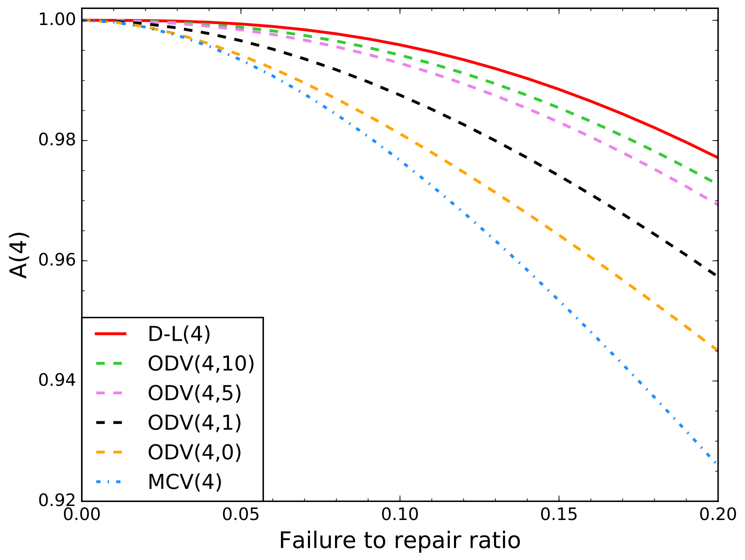

Figure 6: Compared Availabilities for Four Copies

We expected that the introduction of *witness* sites will be a very useful enhancement to this protocol. A witness site is one which holds site availability information and version numbers, but does not hold a replica of the date [11]. A principal advantage of witness copies is that they may be regenerated quickly and at a low cost. Again, the flexibility of our protocol easily accommodates the addition of witness copies. We hope to investigate this in the future.

4.1 Protocol Cost

The number of messages sent by optimistic dynamic voting is the same as the number of messages sent by static majority consensus voting when all sites are active. Each protocol must send a message to each participating site requesting a vote. The number of sites that will respond to this request is the same for both protocols. If we consider the number of messages sent by optimistic dynamic voting over a long period of time, more messages will be sent since optimistic dynamic voting ceases to grant access. The only cost increase in using optimistic dynamic voting over static majority consensus voting is the cost of sending the partition sets and version numbers. This increased cost is minimal, especially since the greatest cost is in generating the message, not the message size.

5 Simulation Analysis

Many difficulties prevent relying solely upon stochastic process modeling: an exponential distribution on repair times is unrealistic, but using other distributions result in intractable problems in most cases; and the problem of modeling network partitions and site failures simultaneously is intractable for all but the most basic cases [20].

We chose to use discrete event simulation to confirm and to extend the results that we obtained using analytic models. By using simulation, we are able to model realistic configuration scenarios involving observed site and network characteristics.

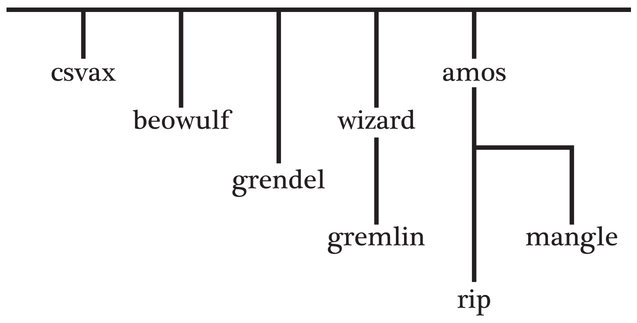

Figure 7: Network Topology

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

An existing network consisting of eight sites and three carrier-sense segments linked by gateways is used as a model. The three subnets are assumed not to fail, this is a realistic assumption for passive media such as carrier-sense segments. The gateway sites may fail resulting in network partitions. Message delivery is guaranteed to all active sites in the current partition when a file access request is made.

Five sites, *csvax, Beowulf, Grendel, Wizard and Amos,* are connected on the main subnet. *Wizard* is the gateway to the graphics laboratory subnet, to which *Gremlin* is connected; *Amos* is the gateway to the linguistics subnet, to which *Rip* and *Mangle* are connected. These sites were chosen as a representative sample of our machines at UCSD. Our main machine, *csvax*, is a VAX-11/780. Due to the heavy load that it carries, it tends to fail often due to software problems. When a hardware failure occurs, repair is rapid since we have spares on site. *Grendel* and *Amos* are VAX-11/750s and do not fail often. When they do fail, it tends to be a hardware failure. Hardware repair characteristics are the same for csvax. All of the VAX machines down for service for three hours every ninety days.

Beowulf is a Pyramid 90x, which tends to fail often due to software failure. When a hardware failure occurs, repair takes longer since spares are not kept on site, but must be brought from out of town. *Wizard, Gremlin, Rip* and *Mangle* are SUNs. Their distinguishing characteristics is that a hardware failure results in at least one week of down time. This is due to our agreement with SUN that requires us to mail the defective part.

Access to the replicated file is modeled as a single user that can access any of the eight sites. The access requests are granted or refused based solely on the current state of the sites containing copies and the protocol's capability to guarantee file consistency. Batch-means analysis was used to compute 95% confidence intervals for all performance indices. All sites were operating at the start of the simulation; the time-to-steady-state interval was taken to be 360 days. A more detailed description of our simulation model can be found in [21].

We considered eight configurations in our study, covering a variety of replica replacement strategies. The first four configurations consist of three replicas; configuration A allows for no partitions, configuration B has a single partition point at *Wizard*, configuration C has partition points at both *Wizard* and *Amos*, and again in configuration D either *Wizard* or *Amos* can cause a partition. The next four configurations consist of four replicas: configuration E allows for no partitions, configuration F has a single partition point at *Wizard*, configuration G has partition points at both *Wizard* and *Amos*, and configuration H has a single partition point at *Amos*.

Table 3 summarizes the unavailabilities of replicated files for all eight configurations and all five consistency policies. Unavailabilities are measured and displayed since they indicate more clearly the differences among the policies.

The performance of Optimistic Dynamic Voting (ODV) was measured assuming no access, one file access per week, one file access per day and one file access per hour. We found that these values span the range of reasonable inter-access times. The measured unavailabilities were expected to fall between those of static Majority Consensus Voting (MCV), which never modifies quorums, and those of instantaneous Dynamic-linear (D-L), where the quorums instantaneously reflect any change in the network status.

When optimistic dynamic voting is employed, even when no accesses occur, the performance is always superior to static majority consensus voting, often performing nearly twice as well. The reason for this is that even though no accesses occur, information about the state of the system is exchanged when a site recovers following a failure allowing some adjustment of quorums. As the access frequency increases, the information available to optimistic dynamic voting becomes more accurate and the availability voting sometimes performs better than its instantaneous counterpart. The reason is that some of the sires such as *csvax*, fail often but repair quickly. Since optimistic dynamic voting only modifies quorum composition when an access occurs, these failures often go unnoticed. This has a buffering effect that prevents optimistic dynamic voting from making a poor choice, which instantaneous dynamic-linear does, such as adjusting the quorum composition include less reliable sites such as the SUNs.

Table 1: Site Characteristics

Site	**Mean Time to Fall (days)**	**Hardware Failures (%)**	**Restart Time (min.)**	**Hardware Repair Time**	
				Constant Part (hours)	**Random Part (hours)**
csvax	36.5	10	20.0	0.0	2
Beowulf	10	10	15	4	24
Grendel	365	90	10	0	2
Wizard	50	50	15	168	168
Amos	365	90	10	0	2
Gremlin	50	50	15	168	168
Rip	50	50	15	168	168
Mangle	50	50	15	168	168

Table 2: Configuration

	csvax	**Beowulf**	**Grendel**	**Wizard**	**Amos**	**Gremlin**	**Rip**	**Mangle**
A	•	•		•				
B	•	•				•		
C	•					•		•
D						•	•	•
E	•	•	•	•				
F	•	•		•		•		
G	•	•				•		•
H	•	•					•	•

Table 3: Replicated File Unavailabilities

Sites	**Consistency Policy**					
	MCV	**D-L**	**ODV**			
			No Access	**Weekly**	**Daily**	**Hourly**
A	0.002130	0.000668	0.001162	0.001012	0.000849	0.000643
B	0.003871	0.001214	0.001872	0.001866	0.001432	0.001161
C	0.031127	0.001707	0.013511	0.010195	0.003492	0.001827
D	0.069342	0.053592	0.064478	0.056692	0.053357	0.051671
E	0.000608	0.000012	0.000269	0.000185	0.000084	0.000017
F	0.002761	0.002154	0.001445	0.001217	0.000947	0.002012
G	0.002027	0.000151	0.000937	0.000608	0.000339	0.000192
H	0.001408	0.000171	0.000764	0.000431	0.000218	0.000101

6 Conclusions

In this paper, we have described our Optimistic Dynamic Voting protocol, and compared it with static majority consensus voting and instantaneous dynamic-linear. Our analysis, augmented by the results of discrete event simulation, has given us interesting results concerning these protocols.

We have found that dynamic voting protocols provide availability that is superior to static majority consensus voting. Our protocol provides this superior performance at the same cost as static majority consensus voting. Our protocol provides the same performance as instantaneous dynamic-linear in the asymptotic case, and quickly converges to it for realistic access rates. Our protocol achieves this at a cost in network traffic similar to that of static majority consensus voting.

We have also found that delaying state information can be of benefit in a real system. Instantaneous dynamic voting protocols can make poor choices, such as shifting the quorum into a less reliable portion of the network, if decisions are made too quickly. Our protocol is more resilient to transient failures; its natural buffering effect occurs because system state information is only propagated when an access occurs.

By using discrete event simulation, we were able to model configuration that we could not model using analytical techniques. The results obtained from our simulation study confirm our predictions made from the Markov analysis. The one surprise was that under certain conditions, optimistic protocols perform better than their instantaneous counterparts. This is due to the buffering effect of maintaining state information only at access time.

Acknowledgments

The authors are grateful to Professor Walter Burkhard and all of the members of the Computer Systems Research Group for their support and encouragement. The first author is especially indebted to Ernestine McKinney and Mary Long for their support.

This work has been done with the aid of Macsyma, a symbolic manipulation program developed at the Massachusetts Institute of Technology. Macsyma is a trademark of Symbolics Incorporated.

References

[1] E. A. Clarence, "Consistency and Correctness of Duplicate Database Systems," *ACM SIGOPS Operating Systems Review*, vol. 11, no. 5, pp. 67–84, 1977.

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

[3] D. K. Gifford, "Weighted Voting for Replicated Data," in *Proceedings of the Seventh ACM Symposium on Operating Systems Principles (SOSP '79)*, pp. 150–162, ACM, 1979.

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

[5] R. H. Thomas, "A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases," *ACM Transactions on Database Systems (TODS)*, vol. 4, no. 2, pp. 180–209, 1979.

[6] D. Davçev and W. A. Burkhard, "Consistency and Recovery Control for Replicated Files," in *Proceedings of the 10th ACM Symposium on Operating Systems Principles (SOSP '85)*, pp. 87–96, 1985.

[7] S. Jajodia and D. Mutchler, "Dynamic Voting," in *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, pp. 227–238, ACM Press, 1987.

[8] S. Jajodia and D. Mutchler, "Enhancements to the Voting Algorithm," in *Proceedings of the 13th Conference on Very Large Databases (VLDB)*, vol. 87, pp. 399–406, 1987.

[9] S. Jajodia, "Managing Replicated Files in Partitioned Distributed Database Systems," in *Proceedings of the Seventh International Conference on Distributed Computing Systems (ICDCS '87)*, pp. 412–418, IEEE, 1987.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] D. D. E. Long and J. F. Pâris, "On Improving the Availability of Replicated Files," in *Proceedings of the Sixth Symposium on Reliability in Distributed Software and Database Systems*, pp. 77–83, 1987.

[15] 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.

[16] H. Garcia-Molina, "Elections in a distributed computing system," *IEEE Transactions on Computers*, vol. 100, no. 1, pp. 48–59, 1982.

[17] H. Garcia-Molina and D. Barbara, "Optimizing the Reliability Provided by Voting Mechanisms," in *Proceedings of the Fourth International Conference on Distributed Computing Systems (ICDCS '84)*, pp. 340–346, 1984.

[18] H. Garcia-Molina and D. Barbara, "How to assign votes in a distributed system," *Journal of the ACM (JACM)*, vol. 32, no. 4, pp. 841–860, 1985.

[19] 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.

[20] K. S. Trivedi, *Probability & Statistics with Reliability, Queuing, and Computer Science Applications.* Englewood Cliffs, New Jersey: Prentice-Hall, 1982.

[21] 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.