Voting with Bystanders

Jehan-François Pâris

Department of Computer Science
University of Houston
Houston, TX 77204-3475

ABSTRACT

Voting protocols ensure the consistency of replicated data objects by disallowing all read and write requests that cannot collect the appropriate quorum of replicas. Voting protocols require a minimum number of three copies to be of any practical use and often disallow a relatively high number of read and write requests.

We present here a voting protocol overcoming this limitation and providing a significant amount of fault-tolerance with as little as two replicas. Voting with Bystanders (VWB), as this protocol is named, applies to all networks consisting of LAN segments that are immune to partial failures linked by gateways that might fail. A stochastic analysis of the protocol under general Markovian assumptions is presented showing that VWB provides excellent read availabilities and good write availabilities with as little as two or three replicas.

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

1. INTRODUCTION

Redundancy has been widely used to increase the fault-tolerance of physical systems. Recent advances in storage and communication technology have extended the scope of the technique to data objects. Critical data are now often replicated to protect them against equipment failures or to reduce read access times. This trend has created a new problem as the multiple copies—or replicas—of a replicated object must be kept consistent. Special consistency protocols have been devised to perform that task and to provide the users with a single consistent view of the object.

Voting protocols [Elli77] probably constitute the best known class of consistency protocols. They ensure the consistency of replicated data objects by disallowing all read and write requests that cannot collect an appropriate quorum of replicas. Different quorums for read and write operations can be defined and different weights, including none, assigned to every replica [Giff79]. Consistency is guaranteed as long as the write quorum W is high enough to disallow parallel writes on two disjoint subsets of replicas, and the read quorum R is high enough to ensure that read and write quorums always intersect.

These conditions are simple to verify, which accounts for the conceptual simplicity and the robustness of voting schemes. Voting has however some disadvantages. It requires a minimum number of three copies to be of any practical use. Even then, quorum requirements tend to disallow a relatively high number of read and update operations.

Several solutions have been proposed to overcome these limitations. Dynamic Voting (DV) [DaBu85] and Dynamic-Linear Voting (DLV) [JaMu87] adjust quorums to reflect changes in replica availability and network topology. Both protocols greatly improve the availability and reliability of replicated objects with more than three replicas. They do not perform better than Majority Consensus Voting (MCV) when only two or three replicas are present. Voting with Witnesses [Pari86a, Pari86b, BMP87] reduces the number of physical copies necessary to achieve a given level of fault-tolerance by replacing some replicas by records of the object status that hold no data. Storage requirements can be significantly reduced but three voting entities are still required to improve upon the availability of non-replicated data. Agrawal and El Abbadi [AgEl88] have recently proposed to store overlapping fragments of the object instead of whole replicas. Their technique significantly reduces storage costs but still requires at least three voting entities.

We present here a voting protocol overcoming this limitation and providing a significant amount of fault-tolerance with as little as two replicas. Voting with Bystanders (VWB), as this protocol is named, applies to all networks consisting of LAN segments that are immune to partial failures linked by gateways that might fail. VWB does not require dynamic adjustments of weights and quorums. Like most consistency protocols, it expressly excludes Byzantine failures.

Section 2 of this paper introduces our protocol. Section 3 presents a brief analysis of replicated object availability under VWB and section 4 discusses several possible extensions to our basic protocol. Finally, section 5 presents our conclusions.

This work was supported in part by a grant from the NCR Corporation and the University of California MICRO program.

2. THE PROTOCOL

VWB extends MCV by generalizing the notion of quorum. Access to the replicated object is given to small groups of replicas that will often contain a single replica. These groups do not need to intersect. Mutual exclusion is enforced by requiring an access quorum that includes (a) one complete group of active replicas and (b) one replica from every group that does not have at least one inactive replica. Accesses that are not mutually exclusive, such as read accesses, require one complete group of active replicas. VWB performs best in environments where it is easy and inexpensive to verify that a replica not answering a quorum call is inactive. This is the case for networks consisting of token rings or CSMA segments linked by gateways or repeaters. Since CSMA segments and token rings are immune to partial communication failures, receiving an acknowledgment from any site on the same token ring or LAN segment as a silent replica is sufficient to establish that the silence of the replica is not due to a network partition and to conclude that the replica must be inactive. The polled sites are called *bystanders* to emphasize the fact that, unlike *witnesses*, they do not know anything about the status of the replicated object.

A more detailed and more accurate description of the protocol follows. We begin by stating the assumptions made about the network and the replicated objects.

2.1. Our Model

A replicated file will consist of a set of replicas residing on distinct sites of a local area network. These sites can fail and can be prevented from exchanging messages by failures of the communication subnet. We will assume that sites not operating correctly will immediately stop operations and that all messages delivered to their destinations will be delivered without alterations in the order they were sent. Byzantine failures are expressly excluded. We will further assume that the communication layer of the network guarantees that two sites that can communicate with the same third site can also communicate with each other.

We will focus our attention on replicated objects that contain uninterpreted values. Two primitive operations on these objects will be defined: a *read* operation that returns the current value of the object and a *write* operation that modifies that value in an arbitrary fashion. To guarantee single-copy serializability, we will disallow concurrent write operations and strictly enforce a single writer policy.

2.2. General Principle

Mutual exclusion between writes is traditionally enforced by requiring all write quorums to intersect. This requirement severely limits the level of fault-tolerance provided by voting protocols as no set of intersecting quorums can allow access to the data in more than one half of the possible replica states [PaBe88]. The Generalized Quorum Consensus protocol (GQC) proposed by Herliyi exploits available type specific properties of data objects to allow more flexible write quorum assignments. This will result in quorums that are better tuned to the requirements of the application and enhance the overall availability of the data. The protocol has however a major limitation. Since it requires that two operations depending on each other have intersecting quorums, any increase in the availability of the data for some operation o_i will necessarily result in a decrease of the availability of the data for all operations that depend on o_i or on which o_i depends.

A better paradigm to follow is the one of Available Copy (AC) protocols [BeGo84, LoPa87]. AC protocols do not rely, like MCV and GQC, on intersecting quorums to guarantee data consistency. They are based on the observation that all live replicas of a data will remain current as long as they receive all write requests. This philosophy has a few important consequences. First, replicated objects remain available as long as at least one replica remains alive. Second, read operations can be satisfied by any live replica. Third, replicas that reside on sites that have failed need to be marked non-available or *comatose* until they are brought up to date. Finally, AC protocols do not guarantee data consistency in the presence of communication failures as it then becomes impossible to guarantee that all live replicas receive all write requests.

We propose here to follow an intermediary approach and to define small subsets of replicas needed to access the data object. We will call a collection of such subsets a *clique* to emphasize the fact that cliques are a generalization of the coteries introduced by Barbara and Garcia-Molina to represent weight assignments in MCV protocols [GaBa85].

Definition 2.1 *A replica is said to be* live *if it resides on a site that has not experienced any failure since the last time the replica was written to or repaired.*

Definition 2.2 *A replica is said to be* dead *if it resides on a site that is not operational.*

Definition 2.3 *A replica is said to be* comatose *if it resides on a site that has failed and the replica has not been repaired yet.*

Definition 2.4 Clique. *Let U be the set of all replicas of a replicated object. A set of groups $C \subset 2^U$ is a* clique *under U iff*

(a) *$H \in C$ implies $H \subseteq U$ and $H \neq \emptyset$, and*

(b) *there are no H, $J \in C$ such that $H \subset J$,*

A *clique S* such that all groups H, $J \in S$ have a non-empty intersection is a *coterie*.

Since the groups of a clique do not necessarily intersect, a process gathering the votes of all the replicas in a group is not guaranteed exclusive access to the replicated object. Mutual exclusion would be guaranteed if every process requesting exclusive access was required to gather the votes of one complete group from the clique plus one vote from every other group in the same clique. This condition is however unnecessarily restrictive as some of these groups may already be unable to compete because they contain one or more dead replicas. Assume for the sake of the argument that the protocol has a mechanism that can reliably certify that a given replica is dead. We could use this mechanism to increase the write availability of the replicated object by excluding from our requirement all groups

containing one or more dead replicas. This solution would however complicate the task of the read protocol as entire groups of replicas could miss a write because one of the replicas was dead when the write took place. This problem disappears once the protocol distinguishes between live replicas and comatose replicas: a group containing a replica recovering from a failure will be identifiable by the fact that the recovering replica will remain comatose until it has been brought up to date.

Rule 1 : Write Rule. *The minimum quorum for a write operation must include:*

(a) *at least one complete group of live replicas from the clique C* **and**

(b) *at least one replica from every group in C that does not contain any replicas that are* dead *or* comatose*.*

The write rule guarantees that, after every write operation, (a) at least one group in *C* only contains active up-to-date replicas, and (b) every group in *C* whose replicas are all active contain at least one up to date replica.

Rule 2 Read Rule. *The minimum quorum for a read operation must include* **either**

(a) *one complete group of live replicas,* **or**

(b) *one replica from every group in the clique.*

Consider for instance a replicated object *X* with two replicas *A* and *B*. The four possible clique assignments are *{{A}}*, *{{B}}*, *{{A*, *B}}*, and *{{A}*, *{B}}*. The three first cliques have all a single subset and constitute valid *coteries*. Cliques *{{A}}* and *{{B}}* represent traditional weight assignments where one of the two replicas is assigned a higher weight than the other one. The clique *{{A*, *B}}* corresponds to a situation in which equal weights are assigned to both replicas. This clique is sub-optimal since it requires both replicas to be available to have a majority. The fourth clique is not a coterie since the groups *{A}* and *{B}* are disjoint. It represents a novel situation in which write requests can be satisfied either by gathering the votes of two live replicas or by obtaining the vote of one live replica and establishing that the other one is dead or comatose. This second alternative illustrates the additional flexibility afforded by our protocol: one group of live replicas is enough to grant access to the replicated object if it is possible to verify that no competing group of live replicas could be formed. Since read requests need to gather the votes of one group of live replicas, they can be satisfied as long as either *A* or *B* is alive.

2.3. The Role of Bystanders

We had assumed in the previous section the existence of a mechanism that can reliably certify that a given replica is dead. Although this mechanism might be difficult or impossible to implement on traditional point-to-point networks, it can be implemented very inexpensively on a large class of local-area networks. Many local-area networks consist of several carrier-sense segments or token rings linked by selective repeaters or gateway hosts. Figure 1 shows one example of such networks: it contains three CSMA segments *AB*, *ACD* and *EF*. *A* is the gateway between *AB* and *ACD* while *ACD* and *EF* are linked by the repeater *X*.

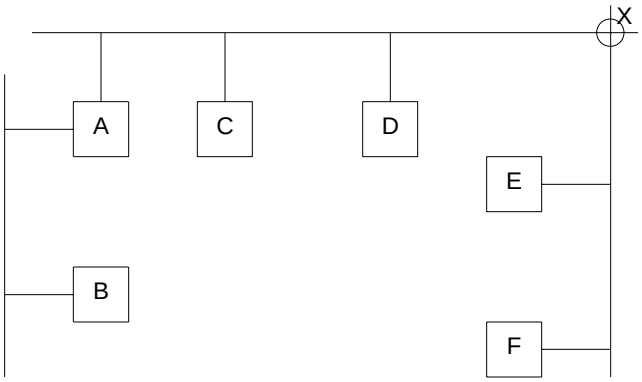

Figure 1: A LAN with Six Sites and Three Segments

Since repeaters and gateways can fail without causing a total network failure, such networks can be partitioned. The key difference with conventional point-to-point networks is that sites that are on the same carrier-sense network or token ring will never be separated by a partition. We will refer to these entities as *LAN segments* [ReTa88]. It is easy in these environments to verify that a replica not answering a quorum is inactive by polling any site residing on the same LAN segment as the site holding the silent replica. One need only to receive one acknowledgment message from any of the polled sites to know that the silence of the replica results from a site failure and not from a network partition. The polled sites are called *bystanders* to emphasize the fact that, unlike *witnesses* they do not know anything about the status of the replicated object. Bystanders have negligible fixed costs and can be implemented on sites that do not have their own secondary storage.

procedure READ(d : data_object)
begin
  let U be the set of all replicas
  $\langle R, \mathbf{v}, \mathbf{s}\rangle \leftarrow$ START(U, d)
  $S_L \leftarrow \{r \in R : r \text{ is } live\}$
  $S_C \leftarrow \{r \in R : r \text{ is } comatose\}$
  $v_{max} = \max_{r \in S_L \cup S_C}\{v_r\}$
  if $(\exists\, c \in C : c \subseteq S_L) \vee (\forall\, c \in C : c \cap (S_L \cup S_C) \neq \emptyset)$ **then**
    perform the read on any $r : v_r = v_{max}$
    COMMIT
  else
    ABORT
  fi
end READ

Figure 2: Read Algorithm

Figures 2, 3 and 4 respectively contain the read, write and recovery algorithms for the VWB protocol. These three algorithms assume that each replica of the data object has a *version number* v_i that represents the ordinal

number of the last successful write recorded in that replica and a *bystander table*. The bystander table has one entry for each site holding a replica of the data object; each entry contains the name of some or all of the other sites residing on the same LAN segment.

The read algorithm starts by broadcasting a message to all replicas requesting their state (*live* or *comatose*) and their version number. It then evaluates S_L, the set of all live replicas, and S_C, the set of all comatose replicas. If the algorithm is able to gather a group of live replicas or one replica from each group $c \in C$, it can assert that the current version number of the replicated object is the maximum version number $v_{\max}$ of all replicas in $S_L \cup S_C$. The read operation can then be performed on any copy in U whose version number is equal to $v_{\max}$.

procedure WRITE
begin
    let U be the set of all replicas
    let B be the set of all bystanders
    let $B(r)$ be the set of all bystanders of replica r
    $\langle R, \mathbf{v}, \mathbf{s} \rangle \leftarrow$ START$(U \cup B,\ d)$
    $S_L \leftarrow \{r \in R : r$ is *live*$\}$
    $S_C \leftarrow \{r \in R : r$ is *comatose*$\}$
    $S_D \leftarrow \{s \in U : s \notin R \wedge B(s) \cap R \neq \emptyset\}$
    $v_{\max} = \max_{r \in S_L \cup S_C}\{v_r\}$
    $T \leftarrow \{s \in S_L \cup S_C : v_s = v_{\max}\}$
    if $(\exists\, c \in C : c \subseteq S_L) \wedge (\forall\, c \in C : c \cap (S_L \cup S_C \cup S_D) \neq \emptyset)$ **then**
        perform the write
        COMMIT$(T, v_{\max}+1)$
    else
        ABORT(R)
    fi
end WRITE

Figure 3: Write Algorithm

The write algorithm starts by broadcasting a message to all replicas requesting their state (*live* or *comatose*) and their version number. A simpler message only requesting an acknowledgment is sent to all bystanders. It then computes the set of live replicas S_L, the set of comatose replicas S_C, and the set of replicas *known to be dead* S_D. It then attempts to establish that S_L includes all replicas from at least one group of C and at least one replica from any group that does not have a replica in $S_C \cup S_D$. If this attempt is successful, the algorithm commits the write operation, sending the new version number to all the copies that were updated.

The recovery algorithm is similar to the read algorithm but uses a two-phase commit protocol like the write algorithm. It begins by ascertaining whether a read quorum exists. If this quorum exists, it computes the maximum version number $v_{\max}$ and determines if the replica being repaired is up-to-date. If it is not, then it is copied from any up-to-date replica.

procedure RECOVER (I : replica)
begin
    repeat
        let U be the set of all replicas
        $\langle R, \mathbf{v}, \mathbf{s} \rangle \leftarrow$ START$(U,\ d)$
        $S_L \leftarrow \{r \in R : r$ is *live*$\}$
        $S_C \leftarrow \{r \in R : r$ is *comatose*$\}$
        $v_{\max} = \max_{r \in S_L \cup S_C}\{v_r\}$
        if $(\exists\, c \in C : c \subseteq S_L) \vee (\forall\, c \in C : c \cap (S_L \cup S_C) \neq \emptyset)$ **then**
            if $v_I < v_{\max}$ **then**
                repair I from any $r : v_r = v_{\max}$
            fi
            COMMIT
        else
            ABORT
        fi
    until successful
end RECOVER

Figure 4: Recovery Algorithm

The protocol we have sketched is a static protocol in the sense that it does not attempt to adjust cliques to reflect changes in replica status or network partitions. Bystander tables need only to be updated when a replica is migrated from one LAN segment to another one. The only significant additional costs occurred by the protocol result then from the additional messages broadcast by the write algorithm to bystanders to ascertain the status of silent replicas. An obvious optimization of the protocol would consist of waiting for replies from replicas before broadcasting to bystanders. This would considerably reduce the overall message traffic as most sites holding replicas can be expected to be operational most of the time.

3. AVAILABILITY ANALYSIS

In this section we present an analysis of the availability provided by our protocol. Several definitions of the availability of a replicated object have been proposed [Pari86a, JaMu88]. We will assume here that the availability of a replicated data object is the stationary probability of the object being in a state permitting access. $A^o_S(n)$ will denote the availability for an operation o of an object with n replicas managed by the protocol S.

Our model consists of a set of sites with independent failure modes that are connected via a network composed of LAN segments linked by gateways or repeaters. When a site fails, a repair process is immediately initiated at that site. Should several sites fail, the repair process will be performed in parallel on those failed sites. We assume that failures are exponentially distributed with mean failure rate λ, and that repairs are exponentially distributed with mean repair rate μ. The system is assumed to exist in statistical equilibrium and to be characterized by a discrete-state Markov process. No attempt is made to model failures of LAN segments, gateways or repeaters.

The assumptions that we have made are required for a steady-state analysis to be tractable [GBS69]. They have

been made in most recent probabilistic analyses of the availability of replicated data [Pari86a, Pari86b, JaMu87, CLP87]. Purely combinational models that do not require assumptions about failure and repair distributions have been proposed [PNP88, ReTa88] but these models cannot distinguish between live and comatose replicas.

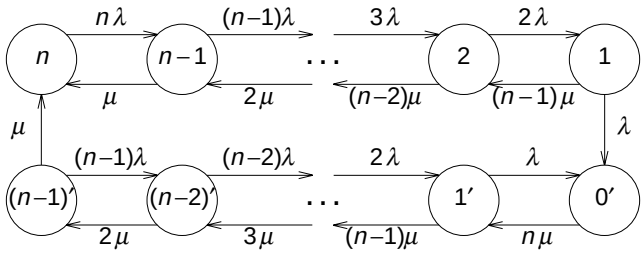

Figure 5: n Replicas on the Same LAN Segment

The easiest topology to analyze has all replicas of the object on the same LAN segment. Since each live replica can act as a bystander for all dead replicas, the best clique for an object with n replicas $r_1, r_2, \cdots, r_n$ is the clique $\{\{r_1\}, \{r_2\}, \cdots, \{r_n\}\}$ whose n groups are all singletons. This clique allows read and write operations as long as one replica remains alive. When that last live replica has failed, the object will remain unavailable until all sites holding replicas have recovered. The protocol behaves identically to a *Naive Available Copy* protocol (NAC) [LoPa87, CLP87] and has exactly the same availability and reliability. Figure 5 contains the state-transition-rate diagram of these two protocols. The n upper states labeled from 1 to n represent the states of the object when 1 to n replicas are alive; the n lower states labeled from 0′ to $(n-1)'$ represent the states of the object when all replicas block have failed and 0 to $n-1$ replicas are on sites that have recovered but remain comatose. Left-to-right and top-to-bottom transitions represent site failures while right-to-left and bottom-to-top transitions indicate site repairs.

The *availability* $A_{VWB}(n)$ of a replicated object with n replicas on the same LAN segment managed by the VWB protocol is then equal to the availability $A_{NAC}(n)$ of the same object managed by the NAC protocol, which is given by:

$$A_{VWB}(n) = A_{NA}(n) = \sum_{k=1}^{n} p_k = \frac{B(n,\rho)}{B(n,\rho) + \rho\, B(n, \frac{1}{\rho})}$$

where

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

and $\rho = \lambda / \mu$ is the failure rate to repair rate ratio.

These availability figures are excellent. In the range of values of ρ found in most installations, they closely approximate the availability afforded by the *Available Copy* protocol, which is the best known protocol for managing replicated objects when network partitions are excluded. Since sites holding replicas double as bystanders, VWB incurs exactly the same message traffic as NAC. All arguments previously made in favor of NAC [CLP87] apply to VWB.

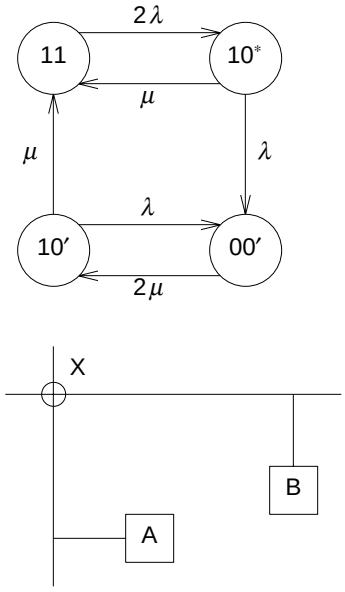

Figure 6: Two Replicas on Two LAN Segments

Consider now the case of a replicated object with *two* replicas *A* and *B* placed on disjoint LAN segments of a network. Assume that the clique $C = \{\{A\}, \{B\}\}$. As long as communication failures are not taken into account, such a system can be represented by a diagram with four states. On the diagram shown in figure 6, state labels represent numbers of operational sites on each LAN segment. States 01 and 01′ are not represented on the diagram as they have been respectively merged with states 10 and 10′. Here too, left-to-right and top-to-bottom transitions represent site failures while right-to-left and bottom-to-top transitions indicate site repairs. The states fall into three categories:

1. *Active States.* The active states are the states in which the file is always accessible for read and write. Active states correspond to situations where all the replicas in at least one group of the clique are live and all groups have at least one live replica. State 11 is the only active state on the diagram of figure 6.
2. *Semi-active States.* The semi-active states are the states in which the file is always accessible for read but relies on the availability of one or more bystanders to allow write accesses. These states correspond to situations where all the replicas in at least one group of the clique are live but some other groups have no live replicas. State 10^* is the only semi-active state on the diagram.
3. *Inactive States* The inactive states are the states in which the data object is inaccessible. States 00′ and 11′ are the two inactive states on the diagram.

The read availability is given by the sum of probabilities of being in any non-primed state:

$$A^R_{VWB}(2) = p_{11} + p^*_{10} = \frac{3\rho + 1}{(\rho + 1)^3}$$

where p_{ij} is the probability of the system being a the state *ij* and $\rho = \lambda / \mu$ is the failure-rate-to-repair-rate ratio. Write availabilities are lower since they depend on the availability of bystanders in all starred states. We have therefore:

$$A^W_{VWB}(2) = p_{11} + p^*_{10} p_b$$

where p_b is the probability that at least one of the bystanders of the dead replica is operational. When each replica has one bystander, we have:

$$p_b = \frac{1}{(\rho + 1)} \quad \text{and} \quad A^W_{VWB}(2,2) = \frac{\rho^2 + 4\rho + 1}{(\rho + 1)^4}.$$

For four bystanders, we have:

$$p_b = \frac{2\rho + 1}{(\rho + 1)^2} \quad \text{and} \quad A^W_{VWB}(2,4) = \frac{\rho^3 + 7\rho^2 + 5\rho + 1}{(\rho + 1)^5}.$$

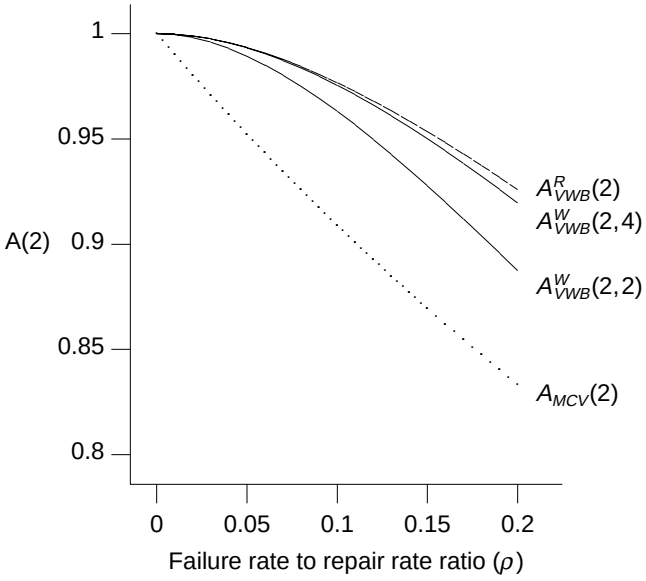

Figure 7: Compared Availabilities for Two Replicas

The graph on figure 7 shows the compared read and write availabilities of MCV with two replicas and VWB with two replicas and two or four bystanders. As expected, MCV with two replicas fails to provide any improvement upon the availability of a non-replicated object. VWB provides an excellent read availability— identical in this case to that afforded by MCV for *three* replicas—and good to very good write availabilities. Since the write availability of an object with four bystanders closely approaches its read availability, adding more bystanders would only have minimal effects on write availability.

A complete analysis of all possible network topologies for more than two replicas would be quite tedious and falls beyond the scope of this paper. We will limit ourselves here to the case of a replicated object with three replicas *A*, *B* and *C* and assume a clique $C = \{\{A\}, \{B\}, \{C\}\}$. The

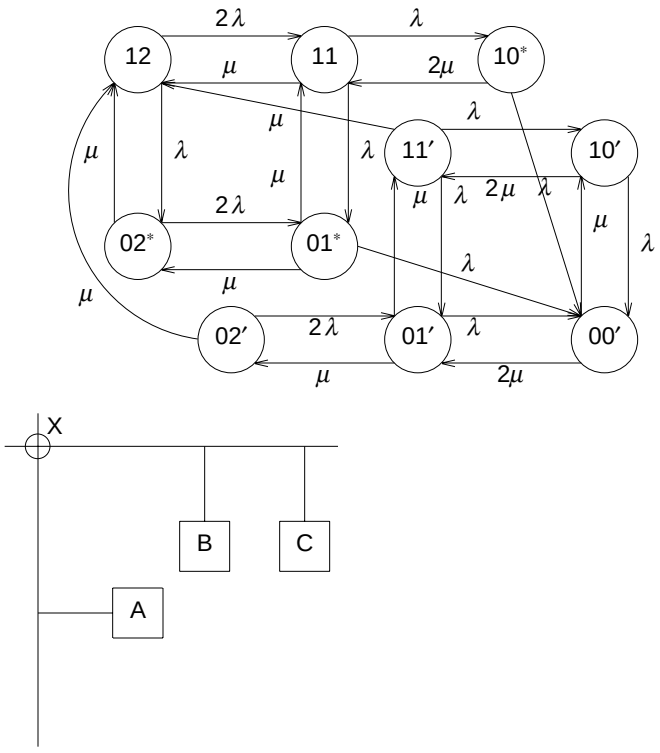

Figure 8: Three Replicas on Two LAN Segments

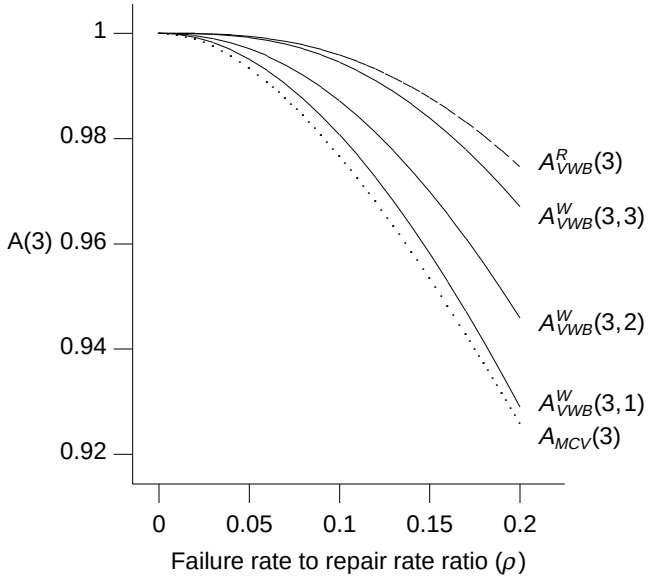

Figure 9: Compared Availabilities for Three Replicas

corresponding state diagram is in figure 8. Here too, state labels represent numbers of operational sites on each LAN segment. The diagram has two active states (12 and 11), three semi-active states (02*, 01* and 10*) and five inactive states (02′, 11′, 01′, 10′ and 00′). The *read availability* of the replicated object $A^R_{VWB}(3)$ is given by the sum of the probabilities of being in a non-primed state:

$$A^R_{VWB}(3) = p_{12} + p_{11} + p^*_{02} + p^*_{10} + p^*_{01}$$

This expression would not be affected if all non-primed states having the same numbers of operational sites—such as 11 and 02^*—were merged. Performing then the same operation on primed states, one could transform the state diagram of figure 8 to the state diagram of figure 5 for $n=3$. Hence $A^R_{VWB}(1+2) = A_{NAC}(3)$. The graph on figure 9 shows the compared read and write availabilities of MCV with three replicas and VWB with three replicas and one, two or three bystanders. VWB provides an excellent read availability and good to very good write availabilities when two or three bystanders are present. Since the write availability of the object for three bystanders closely approaches its read availability, the addition of more bystanders would only have minimal effects on write availability.

4. EXTENSIONS

Two possible extensions of the VWB protocol are briefly discussed here. They concern the applicability of the method to networks with non-transitive communication patterns and the feasibility of dynamic quorum adjustments.

4.1. Non-Transitive Communication Environments

The correctness of the VWB protocol depends on the assumption that the receipt of an acknowledgement message from any bystander of a silent replica is enough to establish that the replica is dead. This assumption is correct as long as the communication layer of the network guarantees that sites that can communicate with the same third site can also communicate with each other.

Gateways between LAN segments can however have their routing tables corrupted by some software error. As a result, the gateway may stop forwarding packets for one or more sites while continuing to handle correctly packets directed to other destinations. It might therefore happen that site *A* can talk to site *B* but not to site *C* while *B* and *C* can communicate with each other [Mull86]. A relatively inexpensive palliative to this problem exists. It consists of replacing the "please-acknowledge" message sent to bystanders of a silent replica by one of the two messages "can you communicate with site *X*?" or "please list the sites with whom you can communicate."

4.2. Dynamic Quorums

The performance of the VWB protocol could be improved by adjusting the clique to reflect changes in replica status or network partitions. Replicas that cannot be reached could be temporarily excluded from the clique until they can be reached again. This would have the advantage of enhancing the write availability of replicated objects by diminishing the reliance of the protocol on bystanders to establish the status of unreachable replicas. We plan to report later on that line of research.

5. CONCLUSIONS

We have presented a voting protocol providing a significant amount of fault-tolerance with as little as two replicas. Voting with Bystanders (VWB), as this protocol is named, applies to replicated objects residing on networks consisting of LAN segments that are immune to partial failures linked by gateways that might fail. Bystanders are sites residing on the same LAN segment as one or more replicas of the object. Bystanders do not know anything about the status of the replicated object and are only used to establish the status of replicas that cannot be reached. A stochastic analysis of the protocol under general Markovian assumptions has shown that VWB provides excellent read availabilities and good write availabilities with as little as two or three replicas and two or three bystanders.

Acknowledgements

This work was supported in part by a grant from the NCR Corporation and the University of California MICRO program. We are grateful to Walter Burkhard, Darrell D. E. Long and all of the members of the Computer Systems Research Group of the CSE Department of the University of California, San Diego for the numerous discussions we had with them at the early stages of our work. We also wish to thank Ms. Elizabeth Mendez for her editorial comments.

The Markov analysis of the availability of the protocols under study has been done with the aid of MACSYMA, a large symbolic manipulation program developed at the Massachusetts Institute of Technology Laboratory for Computer Science. MACSYMA is a trademark of Symbolics, Inc.

References

[AgEl88] D. Agrawal and A. El Abbadi, ‘‘Reducing Storage for Quorum Consensus Algorithms,’’ *Proc. 14th VLDB Conference* (1988), pp. 419-430.

[BeGo84] P. A. Bernstein and N. Goodman, ‘‘An Algorithm for Concurrency Control and Recovery in Replicated Distributed Databases,’’ *ACM Transactions on Database Systems,* Vol. 9, No. 4 (Dec. 1984), pp. 596-615.

[BMP87] W. A. Burkhard, B. E. Martin and J.-F. Pâris, ‘‘The Gemini Replicated File Test-bed.’’ *Proc. 3rd International Conference on Data Engineering* (1987), pp. 596-615.

[CLP87] J. L. Carroll, D. Long and J.-F. Paris, ‘‘Block-Level Consistency of Replicated Files,’’ *Proc. 7th International Conference on Distributed Computing Systems* (1987) pp. 146-153.

[DaBu85] D. Davcev and W. A. Burkhard, ‘‘Consistency and Recovery Control for Replicated Files,’’ *Proc. 10th ACM Symposium on Operating System Principles* (1985) pp. 87-96.

[Elli77] C. A. Ellis, ‘‘Consistency and Correctness of Duplicate Database Systems,’’ *Operating Systems Review,* Vol. 11 (1977).

[GaBa85] H. Garcia-Molina and D. Barbara, ‘‘How to Assign Votes in a Distributed System,’’ *Journal of the Association for Computing Machinery,* Vol. 32, No. 4 (Oct. 1985), pp. 841-855.

[Giff79] D. K. Gifford, ‘‘Weighted Voting for Replicated Data,’’ *Proc. 7th ACM Symposium on Operating*

System Principles (1979), pp. 150-161.

[GBS69] B. V. Gnedenko, , Yu. K. Belyayev and A. D. Soloviev, *Mathematical Methods of Reliability Theory,* (English translation of "Matematicheskiye Metody V Teorii Nadezhnosti"), Academic Press, New York (1969).

[JaMu87] S. Jajodia and D. Mutchler, ‘‘Enhancements to the Voting Algorithm,’’ *Proc. 13th VLDB Conference* (1987), pp. 399-405.

[JaMu88] Jajodia S and Muetchler D, ‘‘Integrating Static and Dynamic Voting Protocols to Enhance File Availability,’’ *Proc. 4th International Conference on Data Engineering* (1988), pp. 144-153.

[LoPa87] D. D. E. Long and J.-F. Pâris, ‘‘On Improving the Availability of Replicated Files,’’ *Proc. 6th Symposium on Reliability in Distributed Software and Database Systems* (1987), pp. 77-83.

[Mull86] K. Muller, private communication.

[PaBe88] J.-F. Pâris and F. Berman, ‘‘How to Make Your Votes Count,’’ submitted for publication.

[Pari86a] J.-F. Pâris, ‘‘Voting with Witnesses: A Consistency Scheme for Replicated Files,’’ *Proc. 6th International Conference on Distributed Computing Systems,* (1986), pp. 606-612.

[Pari86b] J.-F. Pâris, ‘‘Voting with a Variable Number of Copies,’’ *Proc. 16th Fault-Tolerant Computing Symposium* (1986), pp. 50-55.

[PNP88] C. Pu, J. D. Noe and A. Proudfoot, ‘‘Regeneration of Replicated Objects: A Technique and its Eden Implementation,’’ *IEEE Transactions on Software Engineering,* Vol. SE-14, No. 7 (July 1988), pp. 936-945.

[ReTa88] R. van Rennesse and A. Tanenbaum, ‘‘Voting with Ghosts,’’ *Proc. 8th International Conference on Distributed Computing Systems,* (1988), pp. 456-462.

IDIOT