On the Performance of Voting with Ghosts

Jehan-François Pâris Qun Rose Wang *
Department of Computer Science
University of Houston
Houston, TX 77204-3475

Technical Report UH-CS-93-08

Abstract

Voting with Ghosts (VWG) is a static voting protocol tailored to manage replicated data that are stored on networks that can only be partitioned at certain places. We present in this paper the first Markov analysis of the availability of replicated data managed by the VWG protocol and show how this availability can be improved by transforming it into a *dynamic* protocol.

1 Introduction

Data are often replicated in distributed systems either to reduce read access times or to provide constant data availability in the presence of partial network failures. This technique is known as *data replication*. As it can be expected, data replication introduces its own problems, the most important of which is maintaining all replicas in a consistent state. This is a complex task because host failures and network partitions often occasion incomplete updates that leave some replicas untouched.

Special *replication control protocols* have been devised to perform this task in a transparent fashion. These protocols differ in their message overhead, their handling of network partitions and the data availabilities they provide.

A first class of protocols takes the approach that network partitions are unlikely to occasion conflicting updates and that many of these conflicts will be easy to resolve. These are known as *optimistic protocols*. The best known of them is the *available copy protocol* (AC), a variant of which was implemented in the Coda file system [10].

The second class of protocols take the approach that data consistency carries a much more important weight than data availability. These protocols are said to be *pessimistic*. They rely on quorum mechanisms to prevent conflicting updates. As a result, they provide lower data availabilities than optimistic protocols. The best known pessimistic protocols include *majority consensus voting* (MCV), *weighted voting* (WV) [2], *dynamic voting* (DV)[1] and *dynamic-linear voting* (DLV) [3] and *voting with witnesses* (VWW) [7, 4].

Voting with ghosts (VWG) [8, 9] was introduced by Tanenbaum and van Renesse with the twofold intent of (a) providing the level of data availability as optimistic replication control protocols and (b) guaranteeing data consistency in the face of network partitions. VWG is based on the observation that many local area networks consist of indivisible network segments linked by gateways. Hence these gateways are the only places at which these networks can be partitioned. VWG associates with each indivisible network segment a *boot service* whose mission is to monitor the network segment. Every time the boot service detects the failure of a node holding a replica, it starts a *ghost* for that replica. This ghost will hold no state and represent the failed replica for all write quorums.

Tanenbaum and van Renesse found that VWG

*Internet Addresses: paris@cs.uh.edu, rosewang@cs.uh.edu

performed much better than other static voting protocols and nearly as well as the AC protocol. Their findings were unfortunately based on a purely combinational model that did not represent accurately the recovery behavior of the protocol. As a result, there was no way to know how this recovery behavior affected the overall performance of the protocol nor to evaluate the impact of potential refinements.

We present in this paper the first Markov analysis of the availability of replicated data managed by the VWG protocol. Besides the standard Markovian hypotheses, we assume (a) that the boot service is sufficiently replicated to be very unlikely to fail, and (b) that there will always be an operational node on which a ghost can be started. We further use our model to show how the data availability of VWG can be improved by transforming it into a *dynamic voting* protocol.

The remainder of the paper is organized as follows: Section 2 describes the VWG protocol in more detail. Section 3 contains a stochastic analysis of the availability of replicated data managed by VWG. Section 4 introduces our improved dynamic VWG protocol. Finally section 4 has our conclusions.

2 Voting with Ghosts

VWG is a *pessimistic* protocol specifically tailored to networks consisting of *network segments* that cannot be partitioned but are linked by gateways that can fail. Point-to-point networks and other networks that cannot guarantee that some sites will never be separated from each other by a network partition are excluded.

VWG uses *weighted voting* as its basic algorithm. It assigns a number of votes to each copy of a replicated object. Assume the read quorum is r, the write quorum is w and the total number of votes in all copies is N. $r + w > N$ is required because an intersection between the read quorum and the write quorum is needed to guarantee the latest version of the object is contained. Every write operation to a replica of the object includes incrementing the version number v_i associated with the object. If $w \neq N$ and the current version number is unknown, a read quorum is needed to obtain the current version number. If $w = N$ there is no need to maintain a version number since all copies are updated at the same time. At any time, if a read or a write quorum cannot be obtained, the attempted operation has to wait until after a satisfactory number of votes can be reached. Hence no writes are possible if a read *and* a write quorum are not satisfied.

VWG uses *ghosts* to represent failed nodes. A ghost is a process without storage. VWG assumes that there is one *boot service* per network segment whose function is to monitor all nodes holding replicas. Whenever the boot service detects that a node holding a replica has failed, it immediately starts a ghost on an arbitrary node of its network segment. This ghost is assigned the same number of votes as the failed replica.

By definition, network segments cannot partition. Hence, the presence of a ghost replacing a given replica is a guarantee that the replica became unavailable because a site failure and not because a network partition. Ghosts are allowed to participate in write quorums but cannot reply to read quorum requests since they do not have any real storage. Whenever a ghost receives a write request, it pretends to execute the request and answers with a special reply. This special reply is necessary to let the coordinator of the write request verify that there is at least one real replica in the quorum.

When a node recovers from a crash, all replicas residing on that node turn *comatose*. A comatose replica keeps acting like a ghost until (a) it receives a request rewriting the whole replicated object, or (b) it collects a read quorum to copy the latest version of the replicated object. Note that comatose replicas cannot recover whenever the number of operational non-comatose replicas falls beneath the read quorum until *all* replicas have recovered. It is only then that the protocol can compare the version numbers of all replicas and find which ones have the most recent version of the replicated object [8, 9].

3 Markov Analysis

In this section we present an analysis of the availability provided by the voting with ghosts protocol. We define the availability of a replicated object as the stationary probability of the object being in a state permitting access. $A_P^O(n)$ will denote the availability of an object with n replicas managed by a protocol P for an operation O.

Our model consists of a set of nodes with independent failure modes that are connected via an arbitrary network composed of network segments linked by repeaters and gateways. When a node fails, a repair process is immediately started at that node. Should several nodes fail, the repair process will be performed in parallel on those failed nodes. We assume that failures are exponentially distributed with mean failure rate λ; that repairs are exponentially distributed with mean repair rate μ; and that ghost generation process is characterized by a Poisson process with mean rate 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. Similar assumptions have been made in most recent probabilistic analyses of the availability of replicated data [3, 6, 7]. We further assume (a) that the boot service is sufficiently replicated to be very unlikely to fail, and (b) that there will always be an operational node on which a ghost can be started.

We consider first the case where all replicas are on the same indivisible network segment (and data availability is not affected by gateway failures). Figure 1 has the state transition diagram for two replicas located on the same network segment. Each state is identified by a pair $\langle xy \rangle$ where x is the number of operational replicas and y is equal to G if a ghost is present and to 0 otherwise. *Comatose* replicas are identified by an X. Bold arrows represent failure transitions while light arrows denote host recoveries or ghost creations.

State $\langle 20 \rangle$ represents the initial state of the two replicas when they are both operational. A failure of either one of them would bring the system in state $\langle 10 \rangle$; the transition rate from state $\langle 20 \rangle$ to state $\langle 10 \rangle$ is 2λ. The system will remain in state $\langle 10 \rangle$ until (a) the failed replica recovers and the system returns to state $\langle 20 \rangle$, (b) the operational replica fails and the system goes to state $\langle 00 \rangle$, or (c) the boot service creates a ghost to replace the replica that failed and the system moves to state $\langle 1G \rangle$ (one replica and a ghost). The failure of the ghost would return the system to state $\langle 10 \rangle$ while the recovery of the failed replica would bring back the system to state $\langle 20 \rangle$.

Note that there is no state $\langle 0G \rangle$. Since ghosts play no role in the recovery from a failure of *both* replicas, state $\langle 0G \rangle$ is merged into state $\langle 00 \rangle$ and a failure of the operational replica of state $\langle 1G \rangle$ brings the system in state $\langle 00 \rangle$. After a total failure, the recovering replica will stay in a comatose state (state $\langle X0 \rangle$) until both replicas have recovered and the system is back to its original state $\langle 20 \rangle$.

Assuming a *write quorum* $W = 2$ and a *read quorum* $R = 1$, the *read availability* of the two replicas $A_{VWG}^R(2)$ is equal to the sum of the probabilities of being in one of the three states where a read quorum can be obtained, namely $\langle 20 \rangle$, $\langle 10 \rangle$ and $\langle 1G \rangle$:

$$A_{VWG}^R(2) = p_{20} + p_{10} + p_{1G} = \frac{1+3\rho}{(1+\rho)^3}$$

where p_{ij} is the probability for the system being in state $\langle ij \rangle$ and $\rho = \frac{\lambda}{\mu}$ is the failure rate to repair rate ratio. This expression also happens to be the availability of two replicas managed by the *Naive Available Copy* protocol, a variant of the AC protocol that does not maintain any site failure information and waits for the recovery of all replicas after each total failure [6].

The *write availability* of the two replicas $A_{VWG}^W(2)$ is given by

$$A_{VWG}^W(2) = p_{20} + p_{1G} = \frac{1+\psi+3\rho+3\psi\rho+2\rho^2}{(1+\rho)^3(1+\psi+2\rho)}$$

where $\psi = \frac{\kappa}{\mu}$ is the ghost generation rate to repair rate ratio. Note that

$$\lim_{\psi \to 0} A_{VWG}^W(2) = \frac{1}{(1+\rho)^2}$$

while

$$\lim_{\psi \to \infty} A_{VWG}^W(2) = \frac{1+3\rho}{(1+\rho)^3} = A_{VWG}^R(2)$$

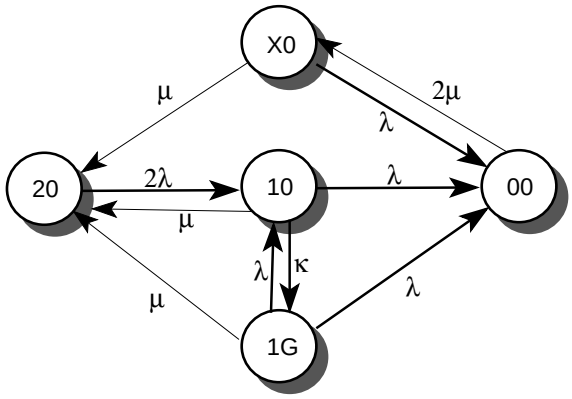

Figure 1: State-transition diagram for two replicas on same network segment

In other words, VWG with two replicas on the same LAN segment provides excellent read and write availabilities whenever the boot process quickly detects replica failures and creates promptly the required ghosts. Failures of the boot service to create these ghosts in a timely fashion has no effect on the read availability of two replicas while it severely limits their write availability. These findings are summarized on the graph of figure 2, where the availability of the two replicas are plotted for values of ρ varying between 0 (no failures) and 0.20 (replicas repair five times faster than they fail and have an average availability of 0.883).

As figure 3 shows, the state transition diagram for two replicas on two network segments linked by a gateway is much more complex since it has to take into account gateway failures. States are represented by triples $\langle xyz \rangle$ where y denotes the state of the gateway (1 if operational, 0 otherwise) and x and z respectively denote the states of the two replicas (1 if operational, 0 if dead, G if replaced by a ghost and X if comatose). Instead of enumerating all the 32 possible states, our diagram merges symmetric states such as $\langle 110 \rangle$ and $\langle 011 \rangle$, which are merged into $\langle 110 \rangle$, and does not keep track of ghosts whenever a read quorum cannot be satisfied. The read availability of the two replicas $A^R_{VWG}(1+1)$ is then given by:

$$A^R_{VWG}(1+1) = \frac{2+11\rho+18\rho^2}{(1+\rho)^2(1+2\rho)(2+3\rho+2\rho^2)}$$

while their write availability is given by:

$$\begin{aligned} A^W_{VWG}(1+1) = & \\ & (\psi^2(4\rho^2+7\rho+2)+\psi(2\rho+1) \\ & (7\rho^2+15\rho+6)+(2\rho+1) \\ & (6\rho^3+13\rho^2+12\rho+4))/ \\ & ((\psi+3\rho+2)(\psi+2\rho+1) \\ & (2\rho^2+3\rho+2)(\rho+1)^3) \end{aligned}$$

Note that

$$\lim_{\psi \to 0} A^W_{VWG}(1+1) = \frac{1}{(1+\rho)^3}$$

while

$$\begin{aligned} \lim_{\psi \to \infty} A^W_{VWG}(1+1) &= \frac{2+7\rho+4\rho^2}{(1+\rho)^3(2+3\rho+2\rho^2)} \\ &\neq A^R_{VWG}(1+1) \end{aligned}$$

As seen on figure 4, we observe here a very different behavior of VWG. While the read availability of the two replicas continues to be quite satisfactory, their write availability is quite low. One can indeed verify that

$$A^W(1+1) < \frac{1}{1+\rho},$$

which is the availability of a *single replica*. In other words, replication actually lowers the write availability of the replicated data. This paradoxical behavior of VWG can be easily explained if one considers that a write quorum must consist of either both replicas or one replica and the ghost of the other one. Hence, any failure of the gateway through which the two replicas communicate will disable all write accesses.

We can generalize this observation to an arbitrary number of replicas and formulate the following theorem:

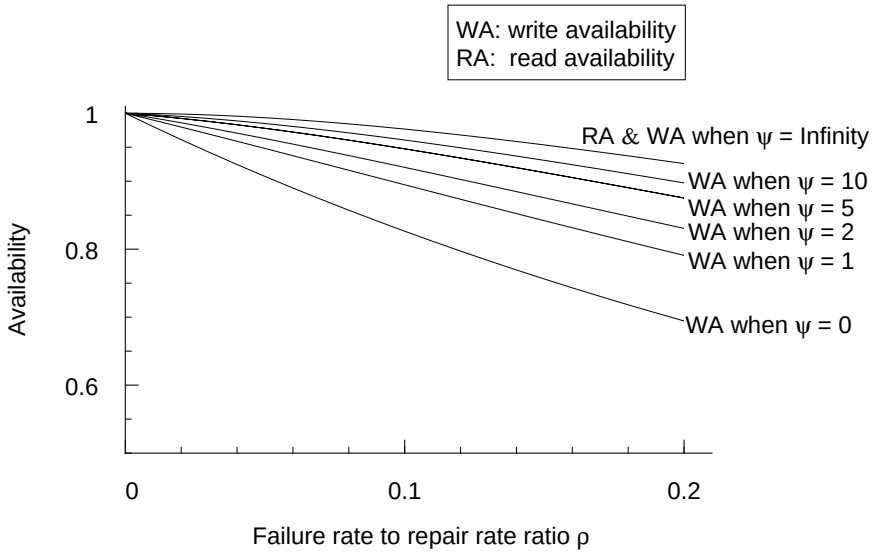

Figure 2: Availabilities of VWG for two replicas on same network segment

Theorem 1 *The* write availability *of replicated data managed by the VWG protocol using a* read one/write all *policy cannot exceed the probability that the replicated data remain unpartitioned.*

Proof: Under a read one/write all policy, the write quorum must include the votes of all nodes holding replicas. While failing nodes can be replaced by ghosts, these ghosts are always created on the same indivisible network segment as the replica they replace. Hence any network partition among the replicas will make the votes of at least one node unavailable and will disable all write accesses as long as the partition has not been repaired.

4 A Dynamic Protocol

The only way to improve the write availability of VWG in environments where network partitions cannot be discounted is to adjust the protocol quorums in such a way that no single gateway failure can prevent the gathering of a write quorum. This necessarily implies the abandonment of the read one/write all policy in favor of smaller write quorums and larger read quorums.

Increasing the size of read quorums has two undesirable consequences. First, it increases the cost of read requests as these requests cannot continue to be handled by a single site. Second, it increases the likelihood that the number of operational non-comatose replicas falls beneath the read quorum. This is a much more serious problem as it appears because of the lengthy recovery process that the event triggers. As we saw earlier, VWG disables all read and write accesses whenever the number of non-comatose replicas falls beneath the read quorum until *all* replicas have recovered and can compare their version numbers.

Consider for instance the case of three replicas managed by VWG with the read and write quorums both set to two. Any simultaneous failure of two of these three replicas would disable all accesses to the replicated data until the three replicas have recovered. Hence, the availability of the replicated data is *lower* than it would have been if the three replicas had been managed by majority consensus voting.

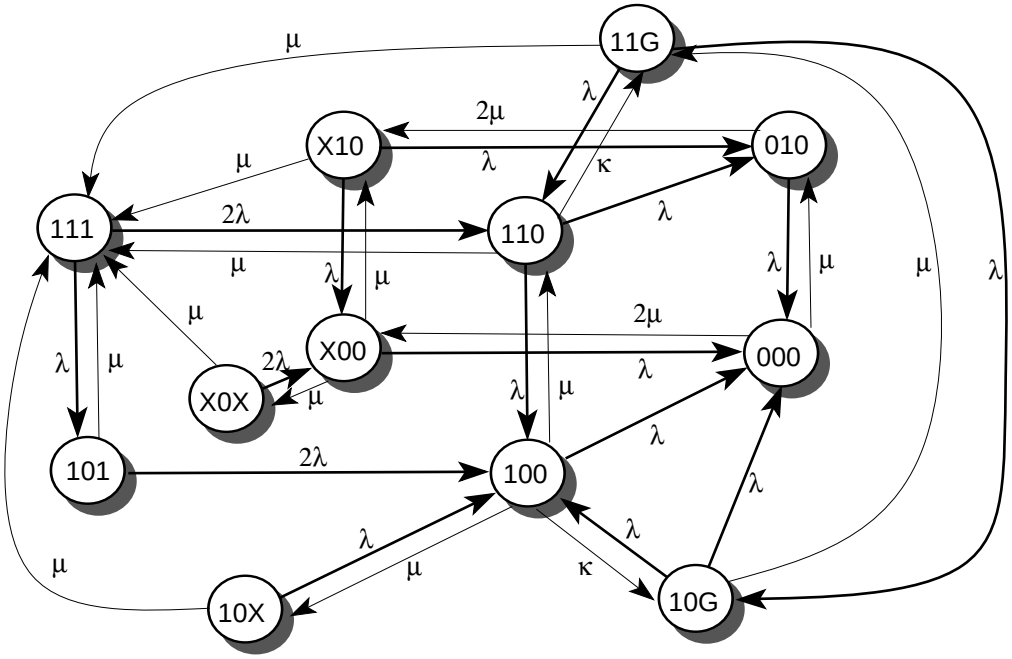

Figure 3: State-transition diagram for two replicas on two network segments

The solution we propose is making the VWG protocol *dynamic*. Dynamic voting protocols adjust read and write quorums whenever they detect a change in the number of available replicas [1, 3, 5]. The main idea is to reduce quorum sizes whenever node crashes and network partitions reduce the number of replicas that can be reached and to increase them whenever a failed replica becomes operational again or a network partition gets repaired. Hence successive quorums adjustments often result in keeping the data available even when a majority of the replicas have become unreachable. Central to all dynamic voting protocols is the notion of a majority partition or *majority block*. This majority block contains all replicas that are believed to be reachable. Whenever the protocol detects that some replicas in the majority block have become unreachable it checks first that a majority of the replicas in the current majority block can be reached. If this is the case, the unreachable replicas are excluded from the majority block and a new majority block is formed. Otherwise the replicated data remain unavailable until some of the unreachable replicas can be reached again. Finally, the excluded replicas are prevented from participating in voting till they are formally reintegrated into the current majority block.

An important issue in the design of any dynamic voting protocol is the mechanism used to detect changes in the composition of the majority block. Davcev and Burkhard's dynamic voting protocol assumed a monitoring process *instantly* recording the state of the network with respect to each site [1]. More recent dynamic voting protocols, such as *dynamic-linear voting* [3] and *optimistic dynamic voting* [5], update the majority block only at access time. These two protocols are much simpler to implement than Davcev and Burkhard's original protocol but fail to react to failures occurring when the replicated data are not accessed. We decided therefore in favor of a third mechanism. In our *dynamic voting with ghosts* (DVWG) protocol, the boot server is responsible for detecting changes in replica accessibility resulting from site failures and network partitions. Whenever this is the case, the boot server that detected the failure immediately re-

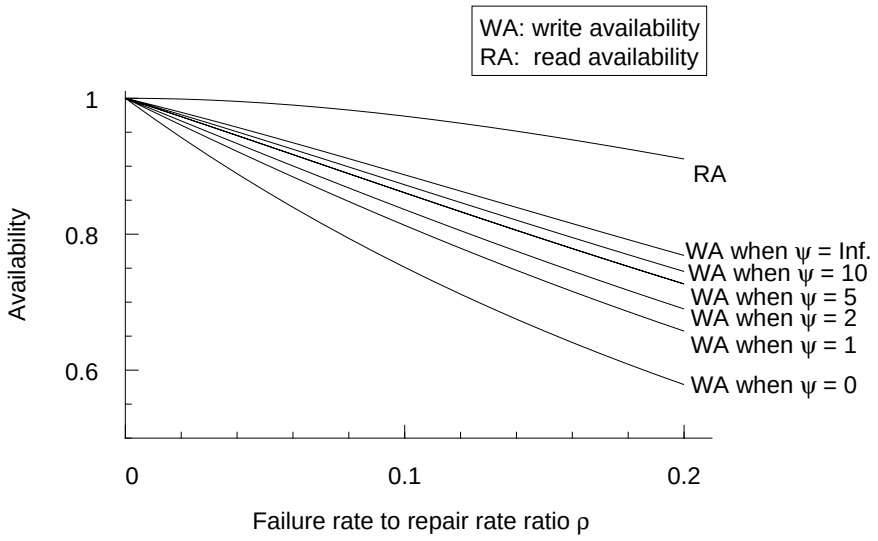

Figure 4: Availabilities of VWG for two replicas on two network segments

quests the formation of a new majority block. Note that there is no need to generate ghosts for the failed replicas unless a new majority block excluding these replicas cannot be formed.

Figure 5 has the state transition diagram for two replicas managed by our DVWG protocol when the two replicas are on the same network segment. States are identified by triples $\langle xyz \rangle$ where x is the number of operational replicas, y the number of ghosts, and z the number of replicas in the current majority block.

State $\langle 202 \rangle$ represents the original state of the system when the two replicas are reachable, the majority block contains the two replicas. A failure of either of these two replicas would bring the system in state $\langle 102 \rangle$. Should the second replica fail before the boot service has detected the failure of the first replica, the system would go to state $\langle 002 \rangle$. A recovery of any of these two replicas would bring the system in state $\langle X02 \rangle$ and the replicated data would remain unavailable until both replicas have recovered and the system brought back to state $\langle 202 \rangle$.

It is however more likely that the boot service will detect the replica failure before both replicas have failed. The boot service will then generate a ghost to replace the failed replica and immediately begin the reorganization of the majority block. This reorganization will succeed because the surviving replica and the ghost of the failed replica constitute valid read and write quorums. As a result of the reorganization, the surviving replica becomes the sole member of the new majority block and the system moves to state $\langle 101 \rangle$. The new read and write quorums are equal to 1. Note that the new state is labeled $\langle 101 \rangle$ and not $\langle 111 \rangle$: since the ghost does not belong to the new majority block, it ceases to play any role in votes.

A failure of the surviving replica would bring the system in state $\langle 001 \rangle$. The recovery of one the two failed replicas would bring the system:

(a) back to state $\langle 101 \rangle$ if the recovering replica is the replica that failed last (and is therefore the sole replica in the current majority block, or

(b) to the new state $\langle X01 \rangle$ if the recovering replica is the other replica.

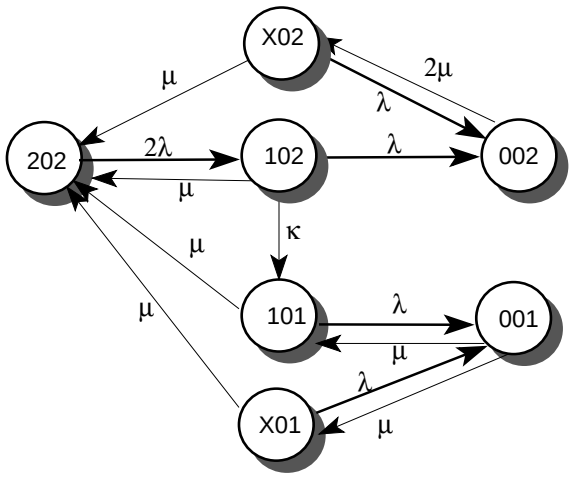

Figure 5: State-transition diagram for two replicas on same network segment

Since state $\langle X01 \rangle$ is an unavailable state. the replicated data will remain until both replicas have recovered and the system brought back to state $\langle 202 \rangle$.

The read availability of the replicated data is given by the sum of the probabilities of the states where a read quorum can be obtained, namely $\langle 202 \rangle$, $\langle 102 \rangle$ and $\langle 101 \rangle$:

$$\begin{aligned} A^R_{DVWG}(2) &= p_{202} + p_{102} + p_{101} \\ &= \frac{1 + 4\rho + 3\rho^2 + \psi(1 + 3\rho + \rho^2)}{(1 + \rho + \psi)(1 + \rho)^3} \end{aligned}$$

where $\psi = \frac{\kappa}{\mu}$. Observing that

$$\lim_{\psi \to 0} A^R_{DVWG}(2) = \frac{1 + 3\rho}{(1 + \rho)^3}$$

and

$$\lim_{\psi \to \infty} A^R_{DVWG}(2) = \frac{1 + 3\rho + \rho^2}{(1 + \rho)^3}$$

one can see that the read availability of two replicas managed by DVWG remains very good for all values of ψ.

The situation is quite different for the *write availability*. We have:

$$\begin{aligned} & A^W_{DVWG}(2) \\ &= p_{202} + p_{101} \\ &= \frac{1 + 3\rho + \rho^2}{(1 + \rho)^3} - \frac{\rho(2 + \rho)}{(1 + \rho + \psi)(1 + \rho)^2} \end{aligned}$$

with

$$\lim_{\psi \to 0} A^W_{DVWG}(2) = \frac{1}{(1 + \rho)^2}$$

and

$$\lim_{\psi \to \infty} A^W_{DVWG}(2) = \frac{1 + 3\rho + \rho^2}{(1 + \rho)^3} = A^R_{DVWG}(2)$$

Figure 6 summarizes these results. While DVWG provides better read and write availabilities than VWG. it still fails to provide acceptable write availabilities whenever the boot process does not detect replica failures in a timely fashion or fails to create the required ghosts.

These availability figures have to be compared with those achieved by Davcev and Burkhard's dynamic voting protocol [1] and by Jajodia and Mutchler's dynamic voting protocol [3]. Since the dynamic voting protocol disallows access to the replicated data whenever a tie occurs, a replicated object with two replicas will remain available as long as *both* replicas are accessible. Hence

$$A^R_{DV}(2) = A^W_{DV}(2) = \frac{1}{(1 + \rho)^2}$$

The availability provided by the dynamic-linear voting protocol is somewhat better since DLV uses the lexicographic ordering of sites to break ties. The protocol therefore distinguishes between a "first" site and a "second" site. Hence the replicated data remain available as long as

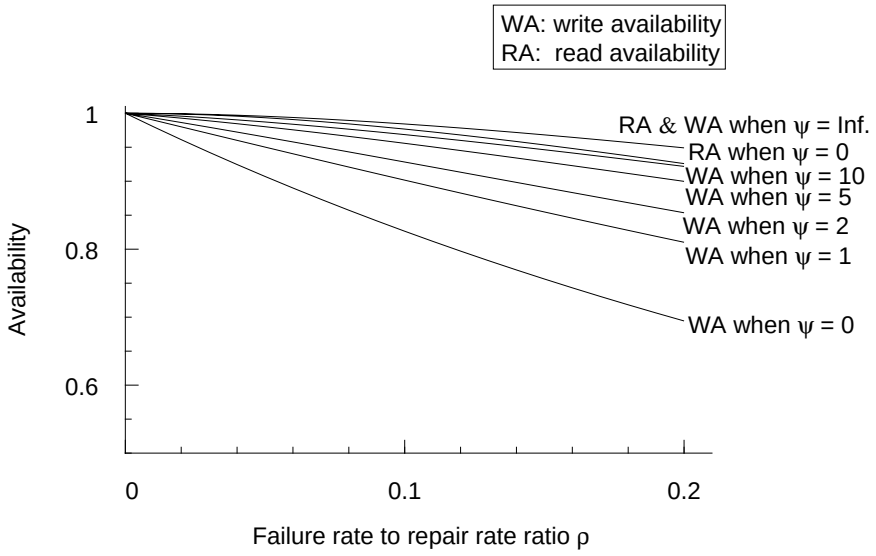

Figure 6: Availabilities of DVWG for two replicas on same network segment

the replica located on the "first" site remain accessible. We have thus

$$A^R_{DLV}(2) = A^W_{DLV}(2) = \frac{1}{(1+\rho)}$$

As seen on figure 7, these availabilities are well below those provided by DVWG as long as the boot server can detect site failures five to ten times faster than sites can be repaired. This condition is very easy to satisfy since the boot server will typically poll sites every few seconds while site repairs may take from ten minutes to several hours or even more. While the ghosts generated by the boot server normally remain short-lived, they play nevertheless an important role as they provide the votes necessary to break ties.

5 Conclusion

VWG is a pessimistic protocol that attempts to replace failed replicas by ghosts residing on the same network segment as the failed replica. Although holding no data, ghosts can testify that the replica they replace have failed and vote for them in all write quorums.

We have presented in this paper the first Markov analysis of the availability of replicated data managed by the VWG protocol under the hypothesis that the ghost creation process was very unlikely to fail. We found that the VWG protocol performed nearly as well as the Naive Available Copy protocol when all the replicas were on the same indivisible network segment. We also found that the VWG provided much lower availabilities when the two replicas it manages were separated by a gateway. Finally, we have shown how the availability afforded by the VWG protocol can be improved by transforming it into a *dynamic* protocol where read and write quorums are adjusted every time the protocol detects a change in the number of reachable replicas.

More work is still needed to study more complex configurations and to evaluate the impact of boot process failures.

References

[1] D. Davcev and W. A. Burkhard, "Consistency and Recovery Control for Replicated Files,"

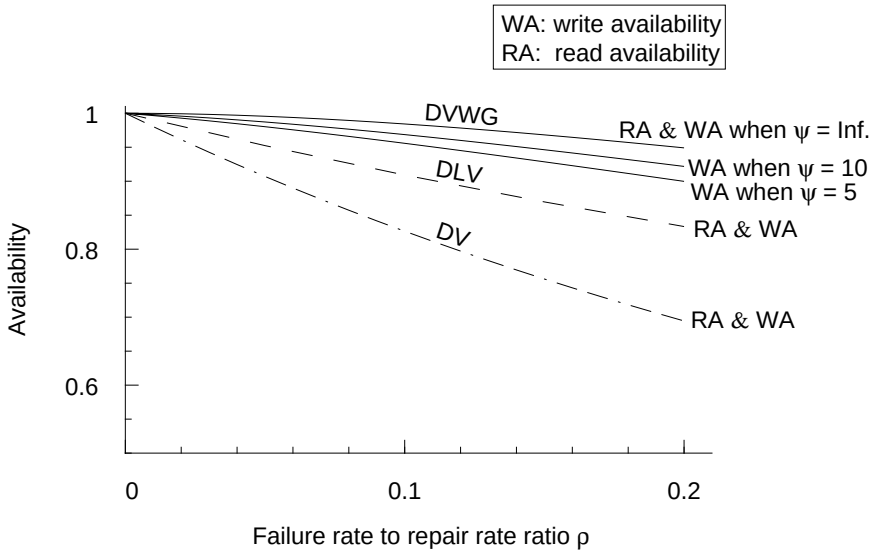

Figure 7: Compared availabilities of DVWG and DLV and DV for two replicas on same network segment

Proc. 10th ACM Symposium on Operating System Principles, (1985) pp. 87-96.

[2] D. K. Gifford, "Weighted Voting for Replicated Data," *Proc. 7th ACM Symposium on Operating System Principles*, (1979), pp. 150-161.

[3] S. Jajodia and D. Mutchler, "Dynamic Voting Algorithms for Maintaining the Consistency of a Replicated Database," *ACM Trans. on Database Systems*, Vol. 15, No. 2 (1990), pp. 230-405.

[4] B. Liskov, S. Ghemawat, R. Gruber, P. Johnson, L. Shriura and M. Williams, "Replication in the Harp File System," *Proc. 13th ACM Symposium on Operating System Principles*, (1991) pp. 226-238.

[5] D. D. E. Long and J.-F. Pâris, "A Realistic Evaluation of Optimistic Dynamic Voting," *Proc. 7th Symposium on Reliable Distributed Systems,* (1988), pp. 129-137.

[6] J.-F. Pâris and D.D.E. Long, "On the Performance of Available Copy Protocols," *Performance Evaluation*, 11 (1990), pp. 9-30.

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

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

[9] R. van Renesse, "The Functional Processing Model," Doctoral Dissertation, Fakulteit der Wiskunde en Informatika, Vrije Universiteit Amsterdam, The Netherlands (1989).

[10] M. Satyanarayanan, J. J. Kistler, P. Kumar, M. E. Okasaki, E. H. Siegel, and D. C. Steere, "Coda: A Highly Available File System for a Workstation Environment," *IEEE Trans. on Computers*, Vol. C-39, 4 (1990), pp. 447-459.