A Highly Available Replication Control Protocol Using Volatile Witnesses

Jehan-François Pâris

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

ABSTRACT

We propose a highly available replication control protocol tailored to environments where network partitions are always the result of a gateway failure. Our protocol divides nodes holding replicas into *local nodes* that can communicate directly with each other and *non-local nodes* that communicate with other nodes through one or more gateways. While local nodes are assumed to remain up to date as long as they don't crash, non-local nodes are required to maintain a volatile witness on the same network segment as the local nodes and must poll this witness before answering any user request. To speed up recovery from a total failure, each site maintains a list of replicas that were available the last time the data were updated or a replica recovered from a crash.

Markov models are used to compare the performance of our protocol with that of the dynamic-linear voting protocol (DLV), which is the best replication control protocol tolerating communication failures. We also observe that volatile witness placement has a strong impact on data availability and gateway nodes are the best location for them.

Keywords: *replicated data, data consistency, replication control protocols, available copy protocols, witnesses.*

1. INTRODUCTION

Many distributed systems maintain multiple copies—or *replicas*—of some critical data to increase the availability of the replicated data and to reduce read access times. Managing multiple replicas of the same data presents however a special challenge as site failures and network partitions are likely to occasion inconsistent updates. Special *replication control protocols* have been devised to avoid this occurrence and guarantee that a consistent view of the replicated data is always presented to their users.

An ideal replication control protocol should satisfy two criteria: First, it should not require more than two replicas to provide a satisfactory data availability. Second, it should provide inexpensive reads. There are protocols meeting these two criterias. They are the so-called *available copy* (AC) protocols. They provide an excellent data availability with just two replicas [2, 7] and guarantee that all available replicas always remain up to date. Hence data can then be read from any available copy. Unfortunately, AC protocols do not guarantee the consistency of the replicated data in the presence of network partitions. AC protocols are said to be *optimistic* because they implicitly assume that inconsistent updates resulting from network partitions will either be infrequent or easy to resolve. There are many protocols that guarantee the consistency of replicated data across network partitions but these protocols require $2n+1$ servers in order to guarantee access to the replicated data in the presence of n simultaneous server failures [1].

We present here a new replication control protocol tailored to environments where network partitions can only occur at a few well-defined partition points and are always the result of a gateway failure. Our protocol implements the same ''write-all/read-one'' rule as the conventional Available Copy (AC) protocols [1-2]. Unlike other AC protocols, our protocol divides nodes holding replicas into *local nodes* that can communicate directly with each other and *non-local nodes* that communicate with other nodes through one or more gateways. While local nodes are assumed to remain up to date as long as they don't crash, non-local nodes are required to maintain a volatile witness on the same network segment as the local nodes and need to interrogate this witness before answering any user request.

The remainder of this paper is organized as follows. Section 2 reviews relevant work. Section 3 presents our new replication control protocol and section 4 discusses its performance. Section 5 introduces a few variants and, finally, section 6 presents our conclusions.

2. RELEVANT WORK

Our protocol is based on three fundamental concepts, namely available copy protocols, volatile witnesses and the notion of an ''aggregate site.'' We review them briefly.

2.1. Available Copy Protocols

Available copy (AC) protocols [1-2] provide an efficient means for maintaining file consistency when network partitions are known to be impossible. The write rule for all

AC protocols is simple: write to *all* available copies. Since all available copies receive each write request, they are kept in a consistent state: data can then be read from *any* available copy. When a replica recovers after a failure, it can repair from any node holding an available replica. Recovering from a total failure requires finding the node that crashed last and marking its replica available [14].

The original AC protocol [1-2] assumed instantaneous detection of failures and instantaneous propagation of this information. Since then, two protocols that do not rely on these assumptions have been devised [7]. The simpler of these, *naive available copy*, maintains no system state information. The other protocol, *optimistic available copy*, maintains system state information only at write and recovery time. Optimistic available copy only approaches the performance of the original protocol since the failure information may be out-of-date, affecting recovery from total failure. But, as the analysis has shown [7], its performance is nearly indistinguishable from that of the original protocol for typical access rates.

Note that the correctness of all AC protocols depends on the fact that every available replica receives all update requests. We need therefore to guarantee that a node that misses a message because of a buffer overflow will always detect that occurrence and mark its replica unavailable until it can check its version number.

2.2. Volatile Witnesses

Witnesses are very small entities that hold no data but maintain enough information to identify the replicas that contain what it believes to be the most recent version of the data. Conceptually this information could be a *timestamp* containing the time of the latest update. Since it is very hard to synchronize clocks in a distributed system, this timestamp is normally replaced by a *version number*, which is an integer incremented each time the data are updated.

Volatile witnesses are witnesses that do not maintain any kind of stable storage and are entirely stored in volatile memory. Unlike conventional witnesses, these witnesses are likely to be left in an incorrect state after a node failure. They must therefore be prevented from participating to elections until they can read the current status of the replicated file. Because of their very small sizes, volatile witnesses can be quickly regenerated every time they become unavailable [8, 12].

2.3. Aggregate Sites

Aggregate sites were introduced to simplify the evaluation of the availability of replicated data whose replicas reside on networks subject to communication failures [10]. The technique is especially adapted to networks consisting of Ethernets that cannot be partitioned and are linked by gateways that can fail [13]. Figure 1 shows one example of such networks: it contains three network segments *DG*, *GABCH* and *EH*. *G* is the gateway between *DG* and *GABCH* while *H* is the gateway between *GABCH* and *EH*. Since 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 nodes that are on the same network segment will never be separated by a partition.

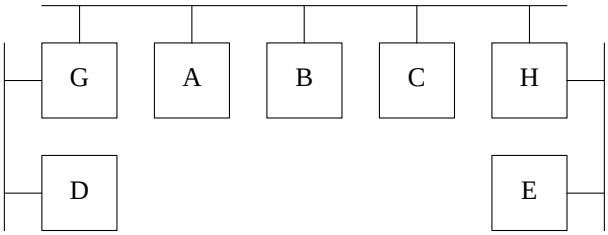


Figure 1: A Network with seven nodes and three segments

Modeling the availability of replicated data having replicas that can be separated by network partitions is a complex task. Hence most studies of replication control protocols neglect network partitions and assume a perfect communication subnet. The aggregate site method operates by selecting first a *backbone* segment which is normally the network segment containing the largest number of replicas. It then replaces all replicas residing on other network segments by aggregate sites that contain the node holding the replica and all the gateways through which the replica communicates with the replicas on the backbone segment. Assuming that the segment *GABCH* was selected as backbone segment, a replica located on node *E* would be replaced by an aggregate site consisting of node *E* and the gateway *H*.

Each aggregate site is assumed to remain operational as long as the node holding the replica is operational and can communicate with the nodes on the backbone segment. This means that the aggregate site $\{E, H\}$ would remain operational as long as *both E* and *H* remain operational. Once all replicas that are not located on the backbone segments are replaced by their aggregate sites, the original network can be replaced by an equivalent *partition-free* network containing the nodes on the backbone segment and the aggregate sites corresponding to all other nodes.

It should be pointed out that the method tends to underestimate the availability of the replicated data because it assumes that non-backbone that cannot communicate with the nodes on the backbone segment do not contribute to the availability of the replicated data. [10]

3. OUR PROTOCOL

Consider again the network of Figure 1 and assume it contains a replicated file *R* with two replicas R_1 and R_2.

Assume that R_1 is located on node *A* while R_2 is located on node *B*. Since the two nodes are on the same network segment *GABCH*, the two replicas can be managed by an AC protocol without risking any inconsistent updates. Should a higher level of data availability be required, a third replica R_3 could be added on node *C*. The replicated data would then remain available as long as at least one of the three nodes holding one of the replicas remains operational.

Locating R_3 on node *D* would have a very different effect. Since *D* is not on segment *GABCH*, a failure of gateway *G* could prevent R_3 from communicating with R_1 and R_2. To guarantee the consistency of the replicated data, we would have to turn to a voting protocol, which would deliver a *lower* data availability than the AC protocol did with the two original replicas. In other words, adding an extra replica on site *D* would actually lower the availability of the replicated data. This would be true for protocols such as Weighted Voting [4], Dynamic Voting [3] and Dynamic-Linear Voting [5] or even Voting with Ghosts [11, 13].

We propose an alternative approach that does not rely on quorum consensus to maintain the data consistent across network partitions. Our protocol divides nodes holding replicas into *local nodes* that can communicate directly with each other and *non-local nodes* that communicate with other nodes through one or more gateways. Local nodes are assumed to remain up to date as long as they are operational since they will directly receive all updates. Non-local nodes may miss updates and are therefore required to maintain a witness on the same network segment as the local nodes. This witness will maintain a *version number* that is incremented every time the replicated file is updated. Whenever a non-local replica wants to verify that it is still up to date, it only needs to compare its own version number with that recorded by the witness.

Our new write rule would thus be: write to *all* available replicas and to *all* available witnesses as long as there is at least one available local replica or one non-local replica and one available witness having identical version numbers. Data can then be read from *any* available local replica or from *any pair* consisting of *one* non-local replica and *one* available witness having identical version number.

3.1. Handling Node Failures

When a node holding a replica recovers from a failure, the replica cannot become available again until it can compare its version number with that of an available local replica or a witness. If the version numbers do not match, it must read the current state of the replicated data from an available local replica or a non-local replica whose version number matches that of one witness. A similar procedure also applies to volatile witnesses: a volatile witness recovering from a node failure will remain unavailable until it can read the current version number from an available local replica or an available witness.

From time to time, it may happen that all nodes holding replicas fail simultaneously. This will somewhat complicate the recovery since Two cases can happen depending on the state of the volatile witnesses:

(a) If there is at least one available witness, recovering replicas can compare their version numbers with that of one of the remaining witnesses. If the two values coincide, the replica is immediately returned to the available state; otherwise it needs to wait for the recovery of other replicas.

(b) If all witnesses have failed, the replicated file will remain unavailable until the replicas that were the last to become unavailable can communicate with each other and compare their version numbers.

The simplest way to find these last available replicas is to wait for the recovery of *all* replicas. This is the strategy used by the naive available copy protocol [7]. While being extremely easy to implement, it does not provide the highest possible data availability. We decided instead to follow the strategy used by the optimistic available copy protocol [7]. We associate with each replica R_i and a *was-available* set listing those copies that received the most recent update. This set includes all replicas that received the most recent write and all replicas that have repaired from R_i since the last write.

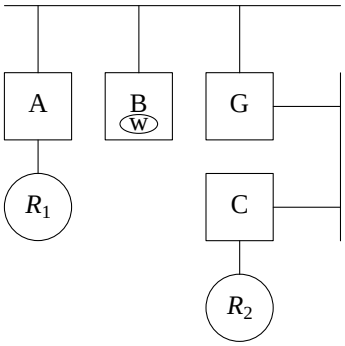
Was-available sets can be maintained inexpensively by ascertaining which replicas are accessible when the replicated file is first accessed and by sending this information along with the first write; the second write will contain the set of replicas which received the first write and so forth. By delaying the information in this way, communication costs are minimized at the expense of some increase in recovery time.

To find the last available replica, we compute the closure of the was-available set with respect to the recovering site R_i The *closure* of a was-available set W_s, written $C^*(W_s)$, is given by:

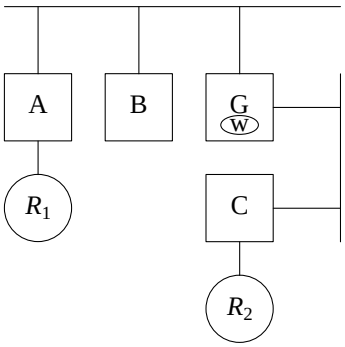
$$C^*(W_s) = \bigcup_{i=0}^{n} C^k(W_s)$$

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

There is an interesting interaction between the method used to maintain was-available sets and our reliance on volatile witnesses. Consider the replicated file represented on Figure 2 (a). It consists of two replicas R_1 and R_2 respectively stored on the servers *A* and *C* and a volatile witness *w* on site *B*. We assume R_1 to be a local replica and R_2 a non-local replica.



(a) Volatile Witness on Node *B*



(b) Volatile Witness on Gateway *G*

Figure 2: Two Replicas on Two Network Segments

Initially R_1, R_2 and w are all available and their three versions numbers are all equal to zero:

A	B	C
$v_A = 0$	$v_B = 0$	$v_C = 0$
$W_A = \{A, C\}$		$W_C = \{A, C\}$

Assume now the following scenario:

(1) node C fails;

A	B	(C)
$v_A = 0$	$v_B = 0$	$(v_C = 0)$
$W_A = \{A, C\}$		$(W_C = \{A, C\})$

(2) node A fails before any other update occurs;

(A)	B	(C)
$(v_A = 0)$	$v_B = 0$	$(v_C = 0)$
$(W_A = \{A, C\})$		$(W_C = \{A, C\})$

(3) node C recovers and its replica becomes available again since its version number matches that of the volatile witness;

(A)	B	C
$(v_A = 0)$	$v_B = 0$	$v_C = 0$
$(W_A = \{A, C\})$		$W_C = \{A, C\}$

(4) replica R_2 and witness w record several updates;

(A)	B	C
$(v_A = 0)$	$v_B = k$	$v_k = k$
$(W_A = \{A, C\})$		$W_C = \{C\}$

(5) nodes B and C both fail;

(A)	(B)	(C)
$(v_A = 0)$	$(v_B = ?)$	$(v_k = k)$
$(W_A = \{A, C\})$		$(W_C = \{C\})$

(6) server A recovers: its replica waits for C recovery.

A	(B)	(C)
$(v_A = 0)$	$(v_B = ?)$	$(v_k = k)$
$(W_A = \{A, C\})$		$(W_C = \{C\})$

If replica R_1 had kept its was-available set updated in real time, it would notice at recovery time that it was the last available replica of the data and would not wait for the recovery of R_2 to mark itself available again. As a result, the data would be left in an inconsistent state. One possible way to avoid this type of occurrence would be

(a) to include witnesses in was-available sets, and

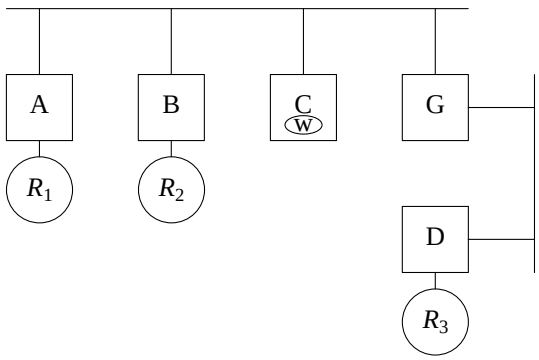
(b) to require witnesses to maintain their own was-available sets and to store them in stable storage.

This would have the obvious disadvantage of forcing the witnesses to become non-volatile.

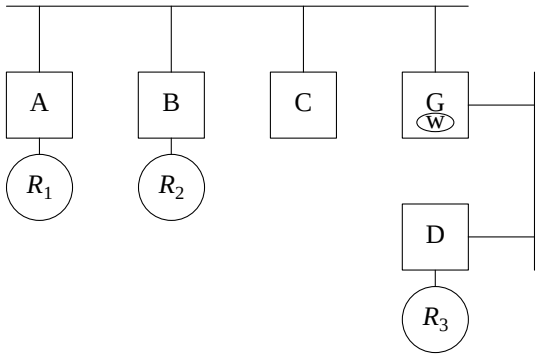
A better approach is to restrict was-available set updates to write times and recovery times as the original optimistic available copy protocol did [7]. Since replicas can only be excluded from an available set because they did not participate to an update, excluded nodes will always be out of date and unable to recover by consulting a volatile witness. ‘‘Dummy updates’’ are not excluded but they must increment the version number of all replicas and witnesses receiving the update.

3.2. A Last Optimization

From time to time, a non-local replica will become the last available replica. Our protocol will detect the situation as soon as the replica gets updated because the was available set of the replica will then become a singleton. Once the protocol has noticed that a non-local replica is the last available replica, it does not need to consult a volatile witness to be sure that the replica has not missed an update.



(a) Volatile Witness on Node *C*



(b) Volatile Witness on Gateway *G*

Figure 3: Three Replicas on Two Network Segments

The non-local replica can then safely become a *temporary local* replica until another replica recovers from it. Until then it can remain operational even if all volatile witnesses have failed. Our write rule then becomes: write to *all* available replicas and to *all* available witnesses as long as there is at least

(a) one available local replica or

(b) one non-local replica and one available witness having identical version numbers or

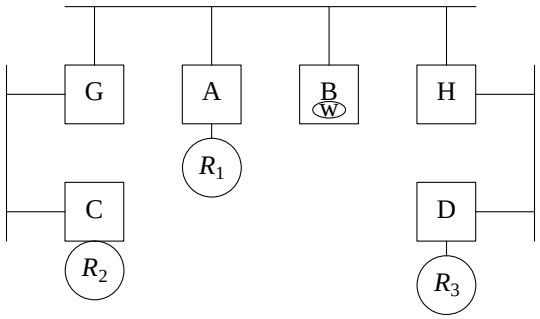
(c) one available replica whose was-available set is a singleton.

Data can then be read from:

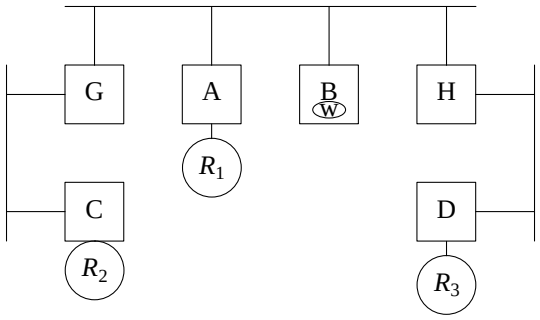
(a) *any* available local replica or

(b) *any pair* consisting of *one* non-local replica and *one* available witness having identical version number or

(c) *any* available replica whose was-available set is a singleton.



(a) Volatile Witness on Node *B*



(b) Volatile Witnesses on Gateways *G* and *H*

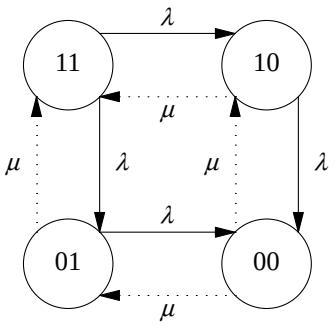
Figure 4: Three Replicas on Three Network Segments

4. AVAILABILITY ANALYSIS

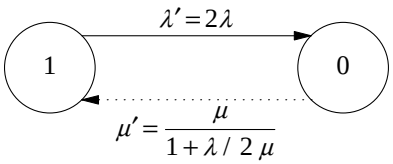
In this section we present an analysis of the availability provided by our protocol and compare it to that afforded by the *dynamic-linear voting*, which is the best pessimistic protocol that does not regenerate witnesses or replicas.

Several definitions of the availability of a replicated data have been proposed [6, 9]. We will assume here that the availability of a replicated file is the stationary probability of the file being in a state permitting access.

Our model consists of a set of nodes with independent failure modes that are connected via a network composed of network segments linked by gateways or repeaters. When a site fails, a repair process is immediately initiated at that site. 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, 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. The assumptions that we have made are required for a steady-



(a) State Transition Diagram for a Node and its Gateway



(b) State Transition Diagram for the Equivalent Aggregate Site

Figure 5: Aggregating a Node with its Gateway

state analysis to be tractable. They have been made in most recent probabilistic analyses of the availability of replicated data [5, 7, 9]. Purely combinational models that do not require assumptions about failure and repair distributions have been proposed [12-13] but these models cannot distinguish between live and comatose replicas.

We will consider six possible replica configurations. They were selected to provide a good sample of all possible configurations with two or three replicas located on two or three different network segments. Configurations where all the replicas are on the same network segment were specifically excluded because our protocol then degenerates into a conventional Available Copy protocol. The two replica configurations represented in Figure 2 consist of two replicas R_1 and R_2 located on two distinct network segments. R_1 is the local replica while R_2 is non-local and needs to access the volatile witness w to check whether it is up to date. Configuration 2(a) has the volatile witness located on one arbitrary node of the network segment *ABG* while configuration 2(b) has the volatile witness located on the gateway G joining the two segments. The two replica configurations of Figure 3 have three replicas on two segments: R_1 and R_2 are local replicas while R_3 is non local. Configurations 3(a) and 3(b) only differ in the location of the volatile witness w. The two configurations of Figure 4 have both three replicas on three network segment. Configuration 4(a) has one volatile witness w on the same segment as the local replica R_1 while configuration 4(b) has one volatile witness on each gateway.

The aggregate site method was used to model the impact of network partitions on the availability of the replicated data. Nodes holding non-local replicas were replaced by equivalent aggregate sites consisting of the node itself and its gateway to the network segment containing the local replicas. This process is illustrated on Figure 5. The state transition diagram 5(a) represents the four possible states for the node and its gateway: they can be both operational (state 11), one of them can be operational while the other is down (states 10 and 01) and both can be down (state 00). Since the aggregate site is operational only when both the node and its gateway are operational, its aggregate failure and repair rates are respectively $\lambda' = 2\lambda$ and $\mu' = \dfrac{\mu}{1 + \lambda / 2\ \mu}$.

While the aggregate site method produces systems of steady-state equations that are tractable by any good symbolic algebra package, these systems of equations tend to be rather large. We will therefore focus here on the simplest configuration under study and present its state transition diagram in some detail. As seen on Figure 6, the state transition diagram for configuration 2(b) has seven states. Each state is identified by a pair $< uv >$ where:

(a) u is the state of the replica R_1, namely be available (1), unavailable (0) or awaiting recovery (X);

(b) v is the state of the aggregate site CG including R_2 (1, 0 or X as for R_1).

Thus state $< 11 >$ represents the state of the replicated data when both replicas are available. Note that the state of the volatile witness is not explicitly represented since that witness resides on the gateway G, and gateway G has been included in the aggregate site CG.

Primed states identify the unavailable states where R_1 is the last available replica while double primed states identify the unavailable states where R_2 was the last available replica. Hence the three available states are $< 11 >$, $< 10 >$ and $< 01 >$.

The availability of the replicated data is then given by

$$A = p_{11} + p_{10} + p_{01}$$

where p_{uv} is the probability that the system is in state $< uv >$.

The availability figures for the six configurations under study are summarized in Figure 7. All these availabilities were computed for values of the failure rate to repair rate ratio $\rho = \lambda / \mu$ varying between 0 and 0.20. The first value corresponds to perfectly reliable sites and the latter to sites that are repaired five times faster than they fail and have an individual availability of 0. 833.

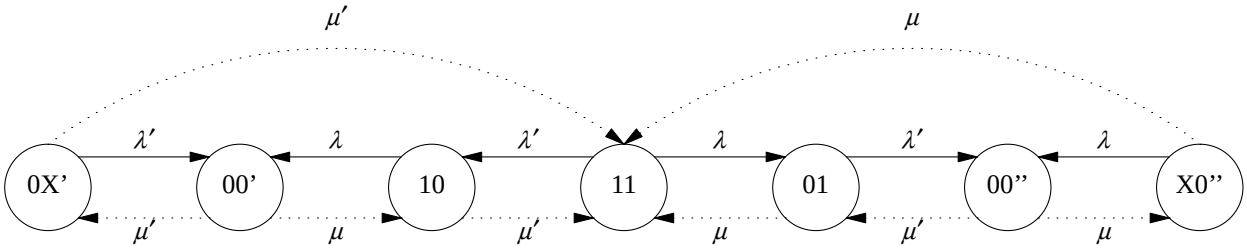


Figure 6: State Transition Diagram for Configuration 2(b)

One will notice first that all three configurations having their witnesses on the gateway, namely configurations 2(b), 3(b) and 4(b) provide much higher availabilities than the corresponding configurations where the witnesses are located on an arbitrary node. The explanation of this phenomenon is simple: whenever a volatile witness is located on the gateway linking a non-local node to the local nodes, it is located within the aggregate site of that node. As a result, the witness will always be operational when the aggregate site (and its replica) are operational. Hence, it appears to the non-local replica as being nearly infallible.

The graph also shows the benefits of as many local replicas as possible. Configurations 3(a) and 3(b), which have two local nodes and one non-local node provides much better data availabilities than configurations 4(a) and 4(b), which have one local node and two non-local nodes.

Our performance study would not be complete without a comparison with extant replication control protocols. We decided to chose the *dynamic-linear voting protocol* (DLV) [5] as benchmark because it is the best pessimistic protocol that does not regenerate failed replicas. The DLV protocol also has the major advantage of providing an upper-bound for the availability of configurations including witnesses: since witnesses do not contain any information that a replica does not contain, we know that the availability afforded by *n* replicas and *m* witnesses will always be bounded by that of a replicated file obtained by replicating the *m* witnesses by *m*r full replicas.

The results of this comparison are displayed on Figure 8, which was obtained by superimposing on Figure 7 the availability curves for a few replica configurations managed by the DLV protocol. All these curves were obtained by the same aggregate site method we used for our AC protocol with witnesses. Configurations managed the DLV protocols are identified by the number of replicas on each network segment; thus $3+1$ identifies a configuration with three replicas on the backbone segment and one replica on another one while $1+2+1$ identifies a configuration with two replicas on the backbone segment and two replicas on two distinct segments.

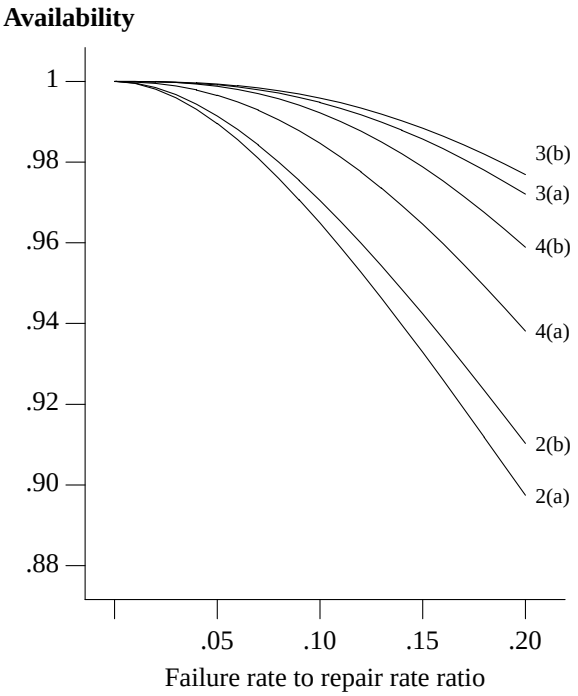


Figure 7: Availabilities of Replicated Data Managed By Our Protocol
(the labels refer to the numbers of the figures 2(a) to 4(b))

The graph emphasizes again the excellent performance of configuration 3(a) and 3(b), which outperform configurations with four replicas, namely $3+1$ and $2+1+1$, managed by DLV. Even when we take into account the cost of the volatile witness, we find that three replicas and one volatile witness outperform four full replicas.

5. EXTENSIONS TO THE ACW PROTOCOL

Many enhancements to our ACW are possible. We briefly discuss two of them.

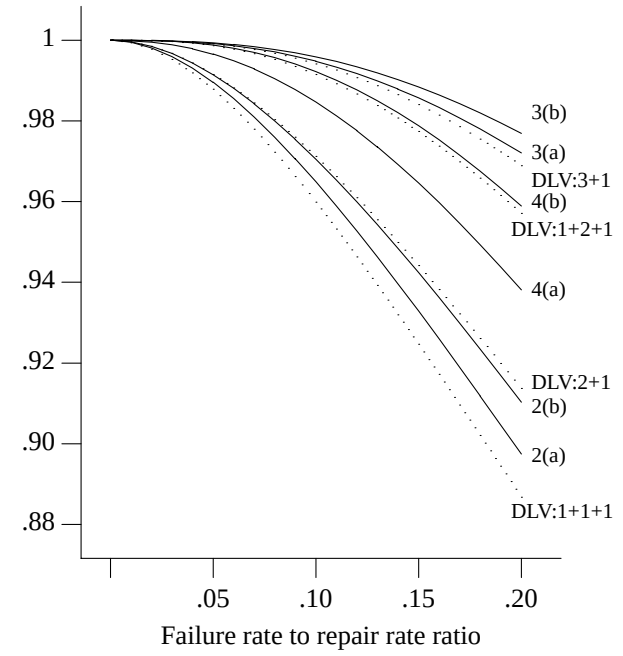


Figure 8: Comparing Our Protocol with Dynamic-Linear Voting

5.1. Replacing Volatile Witnesses by Conventional Witnesses

One way to improve the performance of our protocol would be to replace the volatile witnesses by witnesses that store their version numbers in stable storage. This potential enhancement has the major drawback of excluding all diskless nodes as potential hosts for witnesses. Besides, slightly more robust witnesses are not likely to provide a better data availability than that achieved by locating volatile witnesses on the gateways.

5.2. Implementing Regenerable Witnesses

Another possible approach to increase the resiliency of volatile witnesses would be to replace failed witnesses instead of waiting for the recovery of the failed site. This technique, known as *regeneration* [12], approximates the protection provided by additional witnesses, but at a much lower cost. Here too, the benefits of regeneration are not likely to exceed those that can be achieved by locating volatile witnesses on the gateways.

6. CONCLUSION

We have presented a pessimistic AC protocol tailored to environments where network partitions are always the result of a gateway failure. Our protocol divides nodes holding replicas into *local nodes* that can communicate directly with each other and *non-local nodes* that communicate with other nodes through one or more gateways. While local nodes are assumed to remain up to date as long as they are operational, non-local nodes are required to maintain a volatile witness on the same network segment as the local nodes and need to interrogate this witness before answering any user request. A major advantage of our protocol is the fact that the data will remain available as long as at least one local site or one non-local site and a witness remain available. A preliminary stochastic analysis of the protocol under standard Markov assumptions has indeed confirmed its excellent performance.

REFERENCES

[1] P.A. Bernstein, V. Hadzilakos and V. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison Wesley, Reading, (1987).

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

[3] D. Davcev and W.A. Burkhard, ‘‘Consistency and Recovery Control for Replicated Files,’’ *Proc. 10th ACM SOSP*, (1985) pp. 87-96.

[4] D.K. Gifford, ‘‘Weighted Voting for Replicated Data,’’ *Proc. 7th ACM SOSP*, (1979), pp. 150-161.

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

[6] S. Jajodia and D. Mutchler, ‘‘Integrating static and dynamic voting protocols to enhance file availability,’’ *Proc. 4th ICDE* (1988) pp. 144-153.

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

[8] J.-F. Pâris and D.D.E. Long, ‘‘Voting with Regenerable Volatile Witnesses,’’ *Proc. 7th ICDE*, (1991), pp. 112-119.

[9] J.-F. Pâris, ‘‘Voting with Witnesses: A Consistency Scheme for Replicated Files,’’ *Proc. 6th ICDCS*, (1986), pp. 606-612.

[10] J.-F. Pâris, ‘‘Evaluating the Impact of Network Partitions on Replicated Data Availability,’’ *Proc. 2nd IFIP Working Conf. on Dependable Computing for Critical Applications*, (1991), pp. 28-35.

[11] J.-F. Pâris and Q.R. Wang, ‘‘On the Performance of Voting with Ghosts,’’ *Proc. Int. Symp. on Applied Computer Sciences*, Monnterrey (1993), pp. 75-84 (also available as technical report UH-CS-93-08).

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

[13] R. van Renesse and A. Tanenbaum, ‘‘Voting with Ghosts,’’ *Proc. 8th ICDCS*, (1988), pp. 456-462.

[14] M.D. Skeen, ‘‘Determining the Last Process to Fail,’’ *ACM TOCS*, Vol. 3, No. 1 (1985), pp. 15-30.