Proc. 7th International Conference on Data Engineering (1991), pp. 112-119

Voting with Regenerable Volatile Witnesses

Jehan-François Pâris

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

Darrell D. E. Long

Computer and Information Sciences
University of California
Santa Cruz, CA 95064

ABSTRACT

Voting protocols ensure the consistency of replicated objects by requiring all read and write requests to collect an appropriate quorum of replicas. We propose to replace some of these replicas by *volatile witnesses* that have no data and require no stable storage, and to regenerate them instead of waiting for recovery. The small size of volatile witnesses allows them to be regenerated much easier than full replicas. Regeneration attempts are also much more likely to succeed since volatile witnesses can be stored on diskless sites.

We show that under standard Markovian assumptions two full replicas and one regenerable volatile witness managed by a two-tier dynamic voting protocol provide a higher data availability than three full replicas managed by majority consensus voting or optimistic dynamic voting provided site failures can be detected significantly faster than they can be repaired.

Keywords: distributed file systems, replicated data, voting, witnesses.

1. INTRODUCTION

Fault-tolerance is an important issue in distributed systems owing to the large number of cooperating components. The redundancy present in most distributed systems can be used to enhance fault-tolerance, and the evaluation of this increased resiliency is of great practical interest. The most common measures of fault tolerance is *availability,* which is the steady-state probability that the object is available at any given moment.

While replicating immutable data objects is an easy task, replicating ordinary data objects presents a special challenge as sites and communication failures are likely to result in inconsistent replica updates. Special protocols have been devised to avoid this occurrence. These replication control protocols define a new data abstraction that has the same semantics as an unreplicated instance of the object being replicated while providing a higher resiliency. We refer to this abstraction as a *replicated object*.

The simplest replication control protocol, *majority consensus voting* (MCV), requires a minimum of three replicas to be of any practical use. Even then, quorum requirements disallow a large fraction of read and write operations. Several solutions have been proposed to surmount these limitations. *Dynamic voting* [5] and *dynamic-linear voting* [7] adjust quorums to reflect changes in replica availability and network topology. Both greatly improve availability when applied to replicated objects composed of more than three replicas, but do not perform significantly better than majority consensus voting when fewer than four replicas are present.

Voting with witnesses [4, 12] reduces the number of full replicas necessary to achieve a given level of fault-tolerance by replacing some replicas by *witnesses* that hold no data but can attest to the status of the replicated object. Agrawal and El Abbadi [2] have recently proposed storing overlapping fragments of the object instead of whole replicas. Both of these techniques significantly reduce storage costs but still require at least three voting entities to be kept in stable storage.

One of the authors has recently proposed [14] storing witnesses in volatile memory. Since these *light-weight* or *volatile* witnesses are likely to be left in an incorrect state after a site failure, recovering volatile witnesses must be prevented from participating in elections until they can determine the current status of the replicated object. This can delay recovery of the replicated object, especially when the volatile witnesses are stored at sites subject to frequent failures.

We propose to eliminate this weakness of volatile witnesses by replacing failed witnesses instead of waiting for the recovery of the failed site. This technique, known as *regeneration,* approximates the protection provided by additional witnesses, but at a much lower cost. The minuscule memory requirements of volatile witnesses make this an attractive solution.

Our proposal has several major advantages over other regenerative replication control protocols. First, volatile witnesses can be regenerated inexpensively owing to their small size. Second, attempts to regenerate a volatile witness are more likely to succeed than attempts to regenerate a full replica since diskless sites can be used. Third, regenerating a volatile witness has practically no effect on the network traffic and incurs no significant storage cost. Finally, regenerable volatile witnesses can be combined with two-tier voting to provide a high level of fault-tolerance with as few as two replicas in stable storage. Unlike other replication control protocols that require only two replicas [2, 3, 12, 13, 16], our novel technique requires only two sites providing stable storage, operates correctly in the presence of communication failures, and does not require any knowledge of the network topology.

The remainder of this paper is organized as follows: Section two introduces regenerable volatile witnesses and describes our protocol in detail. Section three contains a stochastic analysis of the availability of the data provided by our protocol. Our conclusions follow in section four.

2. VOLATILE REGENERABLE WITNESSES

The notion of regenerating replicas to replace those lost owing to site failures was first introduced by the *regeneration algorithm* (RA) [15], which provided mutual and serial consistency of replicated objects in a partition-free distributed system. Since then, it has been shown [8, 10, 17] that regeneration is a generally applicable technique that can be profitably combined with many replication control protocols.

In a partitionable distributed system, consensus protocols are required to maintain consistency. Regeneration has been combined with majority consensus voting to obtain a simple protocol for maintaining mutual consistency, and has also been combined with dynamic voting protocols to provide an increased level of fault tolerance [8, 10]. The penalty for using regeneration with these protocols is an increase in both network traffic and peak storage requirements.

A *witness* [12] requires very little storage: only enough so that it can attest to the state of the replicated object. We describe a method that combines regeneration, a two-tier dynamic voting protocol, and volatile witnesses to create a superior protocol for replicated data management. By regenerating only volatile witnesses, both network traffic and storage requirements are significantly reduced while fault tolerance is improved over conventional protocols.

Dynamic Voting with Regenerable Witnesses

Unlike a conventional witness, a volatile witness cannot recover from a site failure until it can obtain the current state of the replicated object from a valid quorum of replicas and witnesses. Using witnesses as full voting entities could result in unstable majorities that would be lost following the failure of the witnesses. To prevent replicated objects from becoming permanently unavailable, the sum of the weights of all replicas must always exceed the sum of the weights of all volatile witnesses. This condition is especially difficult to enforce for dynamic voting protocols since the access quorums are readjusted each time a change in the state of the replicas or the witnesses is detected. To avoid this complexity, our protocol instead uses a two-tier voting scheme where witnesses are solely used to break ties among competing quorums. This has the advantage that all quorums are stable, since they are composed solely of full replicas.

Our two-tier protocol uses a dynamic voting protocol to protect against communication failures. This protocol must satisfy three conditions. First, it must be able to disenfranchise witnesses that belong to previous generations to avoid possible inconsistencies among the replicas. Dynamic voting protocols already provide the necessary mechanism since they must be able to adjust quorums. Second, it must be atomic to ensure consistent creation of witnesses. If this condition is not met, generations with fewer than a quorum can result, causing the replicated object to become permanently inaccessible. We assume that an appropriate commit protocol is used to ensure atomicity. Third, it must take into account that some volatile witnesses may *never* recover completely. This results in our two-tier scheme for using volatile witnesses to break ties.

Our protocol uses sets of site names, called *partition sets,* to record the identity of the replicas and witnesses that constitute a quorum. These partition sets represent the sets of replicas and witnesses that participated in the last successful operation that included the site where it is stored. They are used to determine the required quorum for the next access operation, and are altered when either a read or write operation occurs, and are brought up-to-date when a site recovers from a failure. This is similar to the information required by *optimistic dynamic voting* (ODV) [9]. Partition sets are used by ODV to determine which replicas can form a quorum. For ODV, a quorum is formed either by a majority of the replicas in a current partition set, or by one half of the current replicas with the tie broken according to lexicographic order.

Our protocol requires four pieces of information to be stored along with each replica: an operation number, o_i, and a version number, v_i; P^i_r, the partition set representing the set of replicas that participated in the last operation; and P^i_w, the partition set representing the set of witnesses that participated in the last operation. Witnesses are simpler and only require an operation number to be kept. The version number represents the number of writes that have been applied to the replica of the data object. The operation number is used to determine the most recent quorum and witnesses and can be viewed as a version number for the partition sets.

Regeneration is attempted when a site recovers from a failure and at each access to the replicated object. Frequent accesses ensure a full complement of witnesses and improve the availability of the replicated object. Accomplishing a regeneration requires a majority of the replicas in the previous quorum to be present, or, if there is a tie, some predefined rule to break it. Once a quorum has been formed, spare sites are selected and witnesses are created. The names of the sites holding witnesses are entered into the partition sets P^i_w of each site in the quorum. The operation number o_i of the replicated object is then incremented. Since a quorum must be present, incrementing the operation number has the effect of disenfranchising all replicas and witnesses that did not participate in the regeneration.

Our method is based on the protocol for detecting whether the access request originates within the majority partition. Since there can be only one majority partition, mutual exclusion is guaranteed and consistency is preserved.

Protocol 1. A request for votes is sent to all participating sites, holding either replicas or witnesses, and the following information is collected. R is the set of responding replicas and W is the set of responding witnesses. From each replica in R we obtain o_i, the operation number, v_i, the version number, P_r^i the partition set containing only replicas, and P_w^i the partition set containing only witnesses. From each witness in W we obtain only the operation number o_i.

From these, two important sets are computed: Q_w which is the set of current witnesses, and Q_r, the set of current replicas.

A quorum is present if any of the following four conditions hold.

1. The set of responding replicas constitutes a majority of replicas in the previous quorum.
2. The set of responding replicas contains exactly half of the replicas from the previous quorum and a majority of the current witnesses also respond.
3. The set of responding replicas contains exactly half of the replicas from the previous quorum and exactly half of the current witnesses respond. In this case a quorum exists if the maximal witness in some total ordering of P_w is present.
4. There were no witnesses present in the previous quorum, and exactly one half of the replicas in the previous quorum are present. In this case, a quorum is present if the maximal replica in some total ordering of P_r is present.

The rules given in protocol 1 require some explanation. The key to improving the resiliency of dynamic voting is improving the tie resolution algorithm. A majority of the previous quorum is the most logical choice for the new quorum, and so, as in most dynamic voting protocols, rule (1) declares a majority of replicas to be a quorum. When one half of the previous quorum of replicas is present, a tie will occur and must be resolved according to some predefined rule.

Using volatile witnesses is a better method for breaking ties for two reasons. First, when lexicographic ordering is used a quorum will be formed only 50% of the time. A volatile witness is a better tie breaker than lexicographic ordering as long as its availability remains greater than 0.5. Finally, increasing the regeneration rate or the number of volatile witnesses brings us closer to the goal of an infallible tie breaker.

When there is a tie among the witnesses, rule (3) applies a total ordering on the set of witnesses to determine the quorum. If all else fails, and there are no witnesses then rule (4) resolves the tie by applying a total ordering to the set of replicas. A more precise statement of the quorum conditions is given in figure 1.

The algorithm for performing a read operation is simple. It first determines whether the current partition is the majority partition. A message is sent to all participating sites requesting their partition sets, operation number and version number; those that send replies are considered to be in the current partition. The set of current replicas is found by computing the maximum operation number of all responding replicas. This set of sites holding up-to-date replicas is called the *quorum set.* If the quorum set represents a majority of the previous quorum, represented by P_r^i, then by rule (1) the access request will be granted. If there is a tie, that is exactly one half of the previous quorum are present, the access request will be granted by rule (2) if a majority of the current witnesses are present. If there is a tie both in the number of replicas from the previous quorum and in the number of current witnesses, then by rule (3) the lexicographic orderering on the set of witnesses will be used to break the tie. If there were no witnesses present in the previous quorum, and there is a tie involving the number of replicas from the previous quorum then by rule (4) the tie will be resolved by applying the lexicographic ordering on the set of replicas.

```
function QUORUM ($\mathbf{P}_r, \mathbf{P}_w, R, W, Q_w, Q_r$) : Boolean
begin
  choose any $m \in Q_r$
  if $(|Q_r| > \frac{|P_r^m|}{2}) \vee$
     $(|Q_r| = \frac{|P_r^m|}{2} \wedge |Q_w| > \frac{|P_w^m|}{2}) \vee$
     $(|Q_r| = \frac{|P_r^m|}{2} \wedge |Q_w| = \frac{|P_w^m|}{2} \wedge \max(P_w^m) \in Q_w) \vee$
     $(|Q_r| = \frac{|P_r^m|}{2} \wedge P_w^m = \emptyset \wedge \max(P_r^m) \in Q_r)$ then
     return true
  else
     return false
  fi
end QUORUM
```

Figure 1: Quorum Algorithm

Once it has been determined that the current partition is the majority partition, then access can continue. After the read operation is performed, a check is made to determine if any witnesses have been lost, that is if $|W| < n_w$, where n_w is the initial number of witnesses. Lost witnesses are replaced by locating other sites willing to hold a witness. The operation number is incremented and sent along with the set of current replicas and witnesses, including the newly created volatile witnesses, to all current replicas to serve as their new partition sets. This last action serves to change the quorum requirements for future access requests. Sites holding witnesses store

```
procedure READ
begin
  START
  collect $\langle \mathbf{P}_r, \mathbf{P}_w, \mathbf{o}, \mathbf{v}, R, W \rangle$
  $Q_w = \{r \in W : o_r = \max_{s \in (R \cup W)} \{o_s\}\}$
  $Q_r = \{r \in R : o_r = \max_{s \in (R \cup W)} \{o_s\}\}$
  $S = \{r \in R : v_r = \max_{s \in R} \{v_s\}\}$
  if QUORUM $(\mathbf{P}_r, \mathbf{P}_w, R, W, Q_w, Q_r)$ then
    choose any $m \in Q_r$
    perform the read
    if $|W| < n_w$ then
      generate $T_w$ with $|T_w| = n_w - |W|$
    fi
    COMMIT$(\langle S, W \cup T_w \rangle, o_m+1, v_m, \langle S, W \cup T_w \rangle)$
  else
    ABORT$(R \cup W)$
  fi
end READ
```

Figure 2: Read Algorithm

```
procedure WRITE
begin
  START
  collect $\langle \mathbf{P}_r, \mathbf{P}_w, \mathbf{o}, \mathbf{v}, R, W \rangle$
  $Q_w = \{r \in W : o_r = \max_{s \in (R \cup W)} \{o_s\}\}$
  $Q_r = \{r \in R : o_r = \max_{s \in (R \cup W)} \{o_s\}\}$
  $S = \{r \in R : v_r = \max_{s \in R} \{v_s\}\}$
  if QUORUM $(\mathbf{P}_r, \mathbf{P}_w, R, W, Q_w, Q_r)$ then
    choose any $m \in Q_r$
    perform the write
    if $|W| < n_w$ then
      generate $T_w$ with $|T_w| = n_w - |W|$
    fi
    COMMIT$(\langle S, W \cup T_w \rangle, o_m+1, v_m+1, \langle S, W \cup T_w \rangle)$
  else
    ABORT$(R \cup W)$
  fi
end WRITE
```

Figure 3: Write Algorithm

only the operation number.

There are several items in the algorithm for reading that need explanation. The START operation begins the transaction. R is the set of reachable replicas, and W is the set of reachable witnesses. Four arrays are collected: $\mathbf{o}$, the operation numbers, $\mathbf{v}$, the version numbers, $\mathbf{P}_w$, the partition sets consisting only of witnesses, and $\mathbf{P}_r$, the partition sets consisting only of replicas. The COMMIT operation completes the transaction and transmits the new consistency control information to all current replicas and witnesses.

The algorithm for writing is similar to the algorithm for reading. Again it is ascertained if the current partition is the majority partition by using the QUORUM function. If this is successful, the write operation is performed, followed by the regeneration of any failed witnesses. The operation number o_i and the version number v_i are incremented and sent along with the sets of current replicas and witnesses to all current replicas to serve as their new partition sets.

The recovery algorithm behaves much like the other algorithms, first determining whether the current partition is the majority partition. If the recovering site is able to communicate with the majority partition, then it determines whether the replica at that site is up-to-date by comparing version numbers. If it is not, then it must be copied from one of the replicas in the quorum set. Any failed witnesses are then regenerated. The recovering replica then sends the union of the set of current replicas and itself, and the set of current witnesses, to all current replicas to serve as their new partition sets.

```
procedure RECOVER(k : site identifier)
begin
  repeat
    START
    collect $\langle \mathbf{P}_r, \mathbf{P}_w, \mathbf{o}, \mathbf{v}, R, W \rangle$
    $Q_w = \{r \in W : o_r = \max_{s \in (R \cup W)} \{o_s\}\}$
    $Q_r = \{r \in R : o_r = \max_{s \in (R \cup W)} \{o_s\}\}$
    $S = \{r \in R : v_r = \max_{s \in R} \{v_s\}\}$
    if QUORUM $(\mathbf{P}_r, \mathbf{P}_w, R, W, Q_w, Q_r)$ then
      choose any $m \in Q_r$
      if $v_k < v_m$ then
        copy the object from site $m$
      fi
      if $|W| < n_w$ then
        generate $T_w$ with $|T_w| = n_w - W$
      fi
      COMMIT$(\langle S \cup \{k\}, W \cup T_w \rangle, o_m+1, v_m, \langle S \cup \{k\}, W \cup T_w \rangle)$
    else
      ABORT$(R)$
    fi
  until successful
end RECOVER
```

Figure 4: Recovery Algorithm

3. AVAILABILITY ANALYSIS

In this section we present an analysis of the availability provided by our protocol. We define the availability of a replicated object as the stationary probability of the object being in a state permitting access. $A_P(n, m)$ will denote the availability of an object with n replicas and m witnesses managed by the protocol P.

Our model consists of a set of sites with independent failure modes that are connected via an arbitrary network composed of LAN segments linked by repeaters and gateways. When a site fails, a repair process is immediately started 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, that repairs are exponentially distributed with mean repair rate μ, and that access requests are characterized by a Poisson process with mean κ. The system is assumed to exist in statistical equilibrium and to be characterized by a discrete-state Markov process. No attempt is made to model failures of LAN segments, repeaters or gateways.

The assumptions that we have made are required for a steady-state analysis to be tractable [6]. They have been made in most recent probabilistic analyses of the availability of replicated data [1, 4, 7, 12]. Purely combinational models that do not require assumptions about failure and repair distributions have been proposed [15, 16] but these models cannot distinguish among live replicas that belong to the current majority partition and live replicas that do not.

The availability analysis of a replicated object with witnesses is complicated by the problem of distinguishing between witnesses and full replicas and by taking into account situations where the witnesses and the replicas disagree about the state of the replicated object. We will therefore focus our analysis on the case of a replicated object consisting of *two* full replicas, *one* regenerable volatile witness and an unlimited number of spare sites. Although larger configurations can be analyzed using the same approach, we did not include them here owing to space considerations.

The state transition diagram for a replicated object consisting of two full replicas, one regenerable volatile witness and an unlimited number of spare sites is shown in figure 5. Each state is represented by a tuple $\langle uv/wx \rangle$ where u is the current number of live replicas and v is the number of live witnesses while w and x respectively are the original numbers of replicas and witnesses when the current majority partition was created. States where the replicated object is unavailable due to the lack of a quorum will be identified by a prime mark (′). This will allow us to distinguish between a state as $\langle 11/11 \rangle$, where the live replica and the live witness belong both to the current majority partition, and state as $\langle 11/11 \rangle'$, where the only live replica does not belong to the majority partition and the replicated object is unavailable.

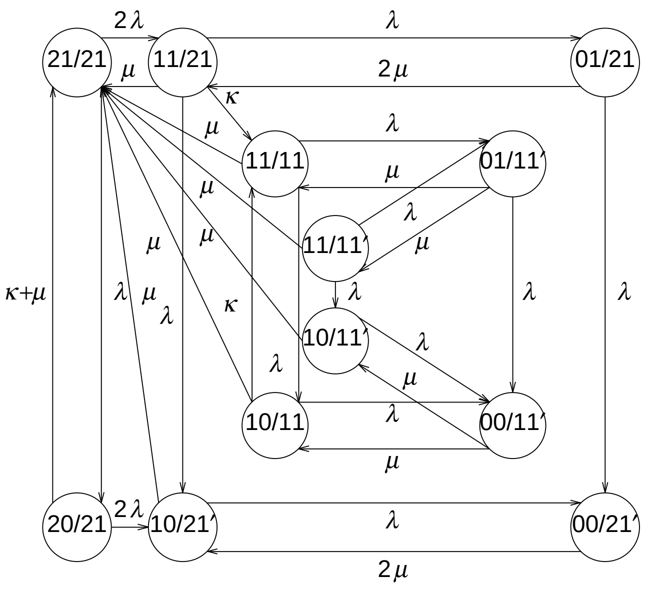

Figure 5: State-Transition Rate Diagram for Two Replicas and One Regenerable Volatile Witness (Optimistic Dynamic Voting)

A replicated object with its two replicas and its witness fully operational is in state $\langle 21/21 \rangle$. A failure of the witness brings the object into state $\langle 20/21 \rangle$ until the failure is detected and the witness is regenerated. If either of the two replicas fails before the failure of the witness is detected, the object enters state $\langle 10/21 \rangle'$ and from there to $\langle 00/21 \rangle'$ if the second replica fails. No witness can be regenerated from state $\langle 10/21 \rangle'$ or state $\langle 00/21 \rangle'$ since no quorum exists in either of these states. A recovery of the failed replica would bring the replicated object directly from state $\langle 10/21 \rangle'$ to state $\langle 21/21 \rangle$ since the recovery process will automatically regenerate the failed witness.

An object in state $\langle 21/21 \rangle$ that loses one of its two replicas enters $\langle 11/21 \rangle$ until the failure is detected. Since the protocol does not regenerate replicas, the object is brought to state $\langle 11/11 \rangle$ and a new majority partition created. An object in state $\langle 11/11 \rangle$ can either recover and return to state $\langle 21/21 \rangle$ or fail again and enter either $\langle 10/11 \rangle$ or $\langle 01/11 \rangle'$. A recovery from $\langle 01/11 \rangle'$ can bring the object either to $\langle 11/11 \rangle$ if the recovering replica belongs to the current majority partition or to $\langle 11/11 \rangle$ if it does not. A failure from state $\langle 10/11 \rangle$ would bring the object to state $\langle 00/11 \rangle'$. Note that there are transitions from $\langle 00/11 \rangle'$ to $\langle 10/11 \rangle$ and to $\langle 10/11 \rangle$ but no transition from $\langle 00/11 \rangle'$ to $\langle 01/11 \rangle'$ since a witness cannot be regenerated by a minority. A failure occurring when the object is in state $\langle 11/21 \rangle$ could bring it to either $\langle 10/21 \rangle'$ or $\langle 01/21 \rangle$. Again there are no transitions from $\langle 10/21 \rangle'$ to $\langle 11/21 \rangle$, nor from $\langle 00/21 \rangle'$ to $\langle 01/21 \rangle$.

The availability of the replicated object is given by the sum of the probabilities of the system being in an available state:

$$A_{RVW}(2,1) = \frac{\rho^2+3\rho+1}{(\rho+1)^3} - \frac{3\rho^4+11\rho^3+10\rho^2}{(3\rho+\phi+1)(\rho^4+7\rho^3+15\rho^2+13\rho+4)} - \frac{4\rho^4+10\rho^3+4\rho^2}{(2\rho^2+\phi(\rho+2)+3\rho+2)(\rho^4+7\rho^3+15\rho^2+13\rho+4)} \quad (1)$$

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

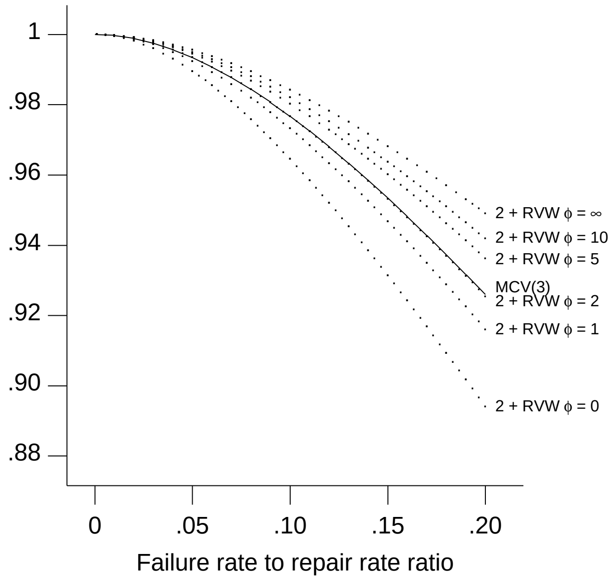

Figure 6. Compared Availabilities of RVW and MCV

The graph in figure 6 shows the compared availabilities for three replicas managed by MCV and two replicas and one regenerable volatile witness managed by our two-tier protocol. As one can see, the availability provided by two replicas and one regenerable volatile witness is strongly affected by the access rate κ and its ratio to the repair rate $\phi = \kappa/\mu$. A replicated object that is frequently accessed ($\kappa > 2\mu$) will have an availability greater than that of three replicas under MCV while a replicated object that is rarely accessed will have a significantly lower availability. This can be inferred from equation (1) by computing

$$\lim_{\phi \to \infty} A_{RVW}(2,1) = \frac{\rho^2+3\rho+1}{(\rho+1)^3},$$

and

$$\lim_{\phi \to 0} A_{RVW}(2,1) = \frac{10\rho^2+11\rho+2}{(\rho+1)(3\rho+1)(2\rho^2+3\rho+2)}.$$

The behavior of a regenerable volatile witness under frequent access conditions is much more interesting. Frequent accesses result in the rapid regeneration of the witness when it fails. The protocol essentially provides a close approximation of a single infallible witness. Assuming independent site failures and neglecting network partitions, a replicated object consisting of *n* replicas and a single infallible witness will remain available as long as at least one of its *n* replicas can be accessed. To recover from a simultaneous failure of all replicas the replicated object will remain unavailable until the replica that failed last recovers. This behavior is identical to that of *n* replicas and no witness managed by the *available copy* protocol. We have therefore for any positive *n*:

$$\lim_{\phi \to \infty} A_{RVW}(n,1) = A_{AC}(n) \quad (2)$$

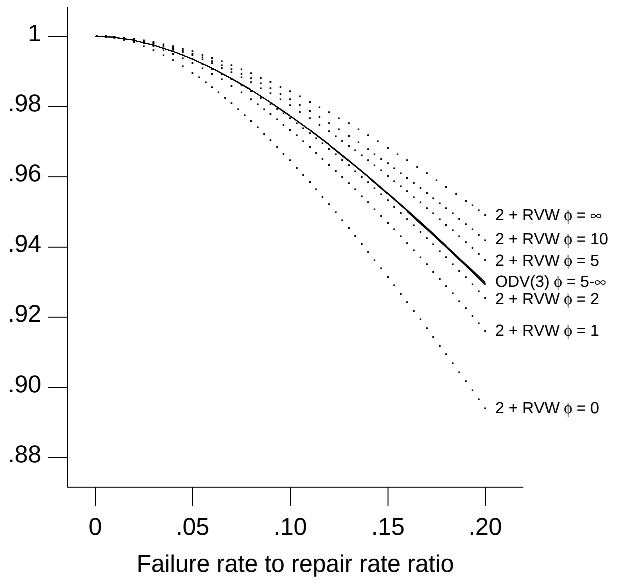

Figure 7. Compared Availabilities of RVW and ODV

This result is significant for several reasons. First, the available copy protocol provides the highest availability of all replication protocols that do not regenerate replicas. Its performance is indeed so close to optimum that it is unlikely that a protocol providing higher availabilities without regenerating failed replicas will be found [11]. Second, the available copy protocol is an idealized protocol that assumes that failures can be instantaneously detected; variants that do not make this assumption, such as naive available copy and optimistic available copy [11], provide somewhat lower data availabilities. Finally, unlike optimistic dynamic voting, the available copy protocol does not guarantee data consistency in the presence of network partitions. Equation (2) states that the cost of this additional protection is one regenerable volatile witness. We have shown in a previous article [11] that *n* replicas managed by the available copy protocol provide a better level of data availability than 2*n*−1 or 2*n* replicas managed by the MCV protocol. Two-tier dynamic voting and a single volatile regenerable witness provides a better level of data availability than MCV with *twice* the number of replicas.

The graph in figure 7 contrasts the availabilities of two full replicas and a volatile regenerable witness with the availabilities of three full replicas managed by ODV. A thick solid line is used to represent the availabilities of three full replicas managed by ODV for values of the access rate to repair rate ratio ϕ between 5 and ∞. Dotted lines are used to represent the availability of two replicas and one regenerable volatile witness for selected values of ϕ between 0 and ∞. The graph again demonstrates the excellent performance of our protocol when the access rate allows a rapid detection of witness failures. This also seems to indicate that very little improvement could be achieved by adding additional volatile witnesses when the access rate to repair rate ratio $\phi \geq 5$.

One might also wonder how our results were affected by our hypothesis that there was an unlimited supply of spare sites ready to host new volatile witnesses. We analyzed a configuration with two full replicas and one regenerable volatile witness where the volatile witness could only be regenerated at one spare site and found that the difference of availability resulting from this restriction was below 2.5% for all values of $\phi \geq 0$ and $0 \leq \rho \leq 0.20$.

A comparison between regenerable volatile witnesses and conventional witnesses would have led to the same conclusions since replacing some replicas of a replicated object by conventional witnesses never increases—and often decreases—its availability. It is also very unlikely that regenerable conventional witnesses would perform significantly better than regenerable volatile witnesses under frequent accesses. First, a single regenerable volatile witness under frequent access already provides an excellent approximation of an infallible witnesses. Second, conventional witnesses can only be regenerated on sites providing stable storage, which greatly reduces the pool of potential regeneration sites.

Since high access rates are essential, one possible implementation of our protocol could maintain all control information on a per site basis. This would guarantee an overall rate of access on the order of every second at least. This solution is especially well suited for replicated devices as each site would have the replicas of a large number of files and directories.

Another interesting observation concerns the need to regenerate witnesses when only one live replica remains accessible. The regeneration transition from $\langle 10/11 \rangle$ to $\langle 11/11 \rangle$ is absolutely useless since the vote of the single live replica in a majority partition is necessary and sufficient to constitute a new majority partition. A bolder protocol would always terminate the witness when the replicated object is found in state $\langle 11/21 \rangle$, which would bring the object in state $\langle 10/10 \rangle$. States $\langle 11/11 \rangle$, $\langle 01/11 \rangle'$ and $\langle 11/11 \rangle'$ would be totally eliminated while states $\langle 10/11 \rangle$, $\langle 00/11 \rangle'$ and $\langle 10/11 \rangle'$ would be respectively renamed $\langle 10/10 \rangle$, $\langle 00/10 \rangle'$ and $\langle 10/10 \rangle'$. For the same reason, we did not contemplate generating new witnesses to replace the replicas that failed.

4. CONCLUSIONS

We have presented a replication control protocol that combines volatile witnesses, regeneration, and a two-tier dynamic voting protocol to provide a higher level of fault-tolerance for a given number of replicas than all extant voting protocols. The excellent performance of our protocol is largely based on the synergy between volatile witnesses and regeneration, as volatile witnesses are quicker and cheaper to regenerate than either full replicas or conventional witnesses.

The behavior of regenerable volatile witnesses under frequent access conditions is the most interesting. Frequent accesses result in the rapid regeneration of failing witnesses. The protocol essentially provides a close approximation of a single infallible witness. Assuming independent site failures and neglecting network partitions, a replicated object consisting of *n* replicas and a single infallible witness will remain available as long as at least one of its *n* replicas can be accessed. Under frequent access, our two-tier protocol using *n* replicas and one regenerable volatile witnesses provides the same availability as the available copy protocol with *n* replicas, and better availability than the majority consensus voting protocol using 2*n* replicas.

We have shown that our protocol provides availability superior to MCV and ODV when the access rate allows a rapid detection of witness failures. We also found that the number of available spare sites has only a small effect on availability. A configuration with two full replicas and one regenerable volatile witness where the volatile witness could only be regenerated at one spare site was found to differ from the case where an unlimited supply of spares was available by less than 2.5% for all values of $\phi \geq 0$ and $0 \leq \rho \leq 0.20$.

More work is needed to assess the reliability of replicated objects using regenerable volatile witnesses, to estimate the impact of communication failures on the availability of the replicated data, and to evaluate the costs incurred by the protocol.

Acknowledgements

We wish to thank Elizabeth Pâris, Mary Long and Richard Golding for their 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

[1] M. Ahamad and M. H. Ammar, ‘‘Performance Characterization of Quorum-Consensus Algorithms for Replicated Data,’’ *IEEE Trans. on Software Engineering,* SE-15, 4 (1989), pp. 492-496.

[2] D. Agrawal and A. El Abbadi, ''Reducing Storage for Quorum Consensus Algorithms," *Proc. 14th VLDB Conf.* (1988), pp. 419-430.

[3] P. A. Bernstein and N. Goodman, ''An Algorithm for Concurrency Control and Recovery in Replicated Distributed Databases,'' *ACM Trans. on Database Systems,* 9, 4 (1984), pp. 596-615.

[4] J. Bechta Dugan and G. Ciardo, ''Stochastic Petri Net Analysis of a Replicated File System,'' *IEEE Trans. on Software Engineering,* SE-15, 4 (1989), pp. 394-401.

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

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

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

[8] D. D. E. Long, J. L. Carroll and K. Stewart, ''Estimating the Reliability of Regeneration-Based Replica Control Protocols,'' *IEEE Trans. on Computers,* TC-38, 12 (1989), pp. 1691-1702.

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

[10] D. D. E. Long and J.-F. Pâris, ''Regeneration Protocols for Replicated Objects,'' *Proc. 5th Int. Conf. on Data Engineering,* (1989), pp. 538-545.

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

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

[13] J.-F. Paris ''Voting with Bystanders,'' *Proc. 9th Int. Conf. on Distributed Computing Systems,* (1989), pp. 394-401.

[14] J.-F. Pâris, ''Efficient Voting Protocols with Witnesses,'' *Proc. 3rd Int. Conf. on Database Theory,* Lecture Notes in Computer Science, Springer Verlag (1990), to appear.

[15] C. Pu, J. D. Noe and A. Proudfoot, ''Regeneration of Replicated Objects: A Technique and its Eden Implementation,'' *IEEE Trans. on Software Engineering,* SE-14, 7 (1988), pp. 936-945.

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

[17] M. Rusinkiewicz and D. Georgakopoulos, ''Maintaining Replicated Data Objects in a Server with a Tunable Degree of Reliability,'' *Proc. 2d Int. Software for Strategic Systems Conference,* (1988), pp.235-240.