Using Volatile Witnesses to Extend the Applicability of Available Copy Protocols

Jehan-François Pâris

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

1. INTRODUCTION

An important consideration in many applications involving data replication is the number of copies or replicas required to achieve a given level of data availability. We would like ideally, to provide continuous access to the replicated data as long as at least one server holding a copy remains accessible. Unfortunately, replication control protocols that achieve this objective do not guarantee the consistency of the replicated data in the presence of network partitions. They 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 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 Available Copy (AC) protocol [1-2]. Unlike the AC protocol, our protocol divides servers holding copies into *local servers* that can communicate directly with each other and *non-local servers* that communicate with other servers through one or more gateways. While replicas stored on local servers are assumed to remain up to date as long as their server remains operational, replicas stored on non-local servers are required to maintain one or more volatile witnesses on the same LAN segment as the local servers and need to interrogate one of these witnesses before answering any user request.

2. THE AVAILABLE COPY PROTOCOL

The Available Copy protocol [1-2] provides a efficient means for maintaining file consistency when network partitions are known to be impossible. The write rule for the 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 server recovers following a failure, it can repair from any server holding an available copy. Recovering from a total failure requires finding the server that crashed last and marking its copy available. The original AC protocol [1-2] assumed instantaneous detection of failures and instantaneous propagation of this information. Since then, variants that do not rely on these assumptions have been devised. One of them, the *naive available copy* (NAC) protocol, does not maintain any state information and waits until all replicas of the replicated object have recovered to ascertain which replica failed last. Another variant, the *optimistic available copy* (OAC) protocol, only maintains state information at write and recovery times. These protocols have been found to perform nearly as well as the original AC protocol, which was found to perform much better than quorum based protocols [7].

3. OUR PROTOCOL

Many local-area networks consist of several carrier-sense segments or token rings linked by selective repeaters or gateway hosts. Figure 1 shows one example of such networks: it contains

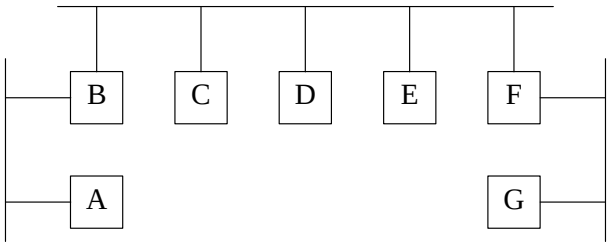

Figure 1: A LAN with seven sites and three segments

three CSMA segments *AB, BCDEF* and *FG*. *B* is the gateway between *AB* and *BCDEF* while *F* is the gateway between *ACDEF* and *FG*. 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 sites that are on the same carrier-sense network or token ring will never be separated by a partition. We will refer to these entities as *LAN segments* [11].

Consider now a replicated data object X with two replicas X_1 and X_2 respectively located on sites *C* and *D*. Since the two sites are on the same LAN segment, the two replicas can be managed by an AC protocol without risking any inconsistent updates. Adding a third replica either at site *A* or at site *G* would have the paradoxical effect of *lowering* the availability of the replicated data if the AC protocol is replaced by a protocol guaranteeing the consistency of the replicated data in the presence of a gateway failure. This would be true for protocols such as Weighted Voting [5], Dynamic Voting [3-4, 7], Dynamic-Linear Voting [6], Voting with Ghosts [11] or Voting with Bystanders [9].

We propose an alternative approach that does not rely on quorum consensus to maintain the data consistent across network partitions. Our protocol divides servers holding replicas into *local servers* that can communicate directly with each other and *non-local servers* that communicate with other servers through one or more gateways. Since copies stored on local servers receive directly all updates, they will remain up to data as long as their server remains operational. This is not true for copies stored on non-local servers as gateway failures may result in lost updates. We require therefore copies stored on a non-local server to maintain one or more witnesses on the same LAN segment as the local servers. These witnesses will maintain a count of number of successful writes to the replicated object (*version number*). Whenever a copy stored on a non-local server wants to verify that it is still up to date, it only needs to compare its own version number with that recorded by one of these witnesses. Witnesses recovering from a site failure must remain silent—or comatose—until they repair from an available witness or an available local site. Hence, they can be stored in volatile storage without any loss of performance.

Our modified write rule would thus be: *write to all available copies and to all available witnesses*. Data can then be read from *any* available local copy or from any pair consisting of a non-local copy and an available witness having the same version number. When a server recovers following a failure, it can repair its copy from any local server holding an available copy or from any non-local server holding a copy whose version number matches that of one witness.

From time to time, it may happen that all servers holding a copy of a replicated data object fail simultaneously. As a result, there will be no available copies of the replicated object. Two cases can happen depending on the state of the volatile witnesses.

(1) If there is at least one witness for the replicated data object that has remained operational, servers that recover can compare the version number of their copy with that of one of the remaining witnesses. If the two values coincide, the server can immediately mark its copy as being available; otherwise it needs to wait for the recovery of other replicas. not up

(2) If all witnesses for the replicated data object have failed, the replicated data object will remain unavailable until all copies that are known to be the last available copies of the replicated data object can communicate with each other and compare their version numbers.

The simplest way to find which copies are the last known available copies of a replicated data object is

to associate with each copy r_i of the object 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 operational when the replicated data object 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. Available copy protocols using was-available set are said to be *optimistic* because they rely on somewhat out-of-date system state information and make the implicit assumption that the few discrepancies that might occur between the true system state and the information they maintain is not likely to affect the overall duration of the recovery process in a significant fashion [7].

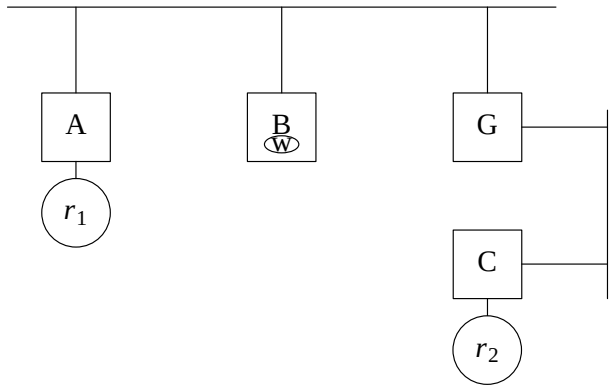

Figure 2: An Object with Two Copies and One Volatile Witness

One should mention an interesting interaction between the method used to maintain was-available sets and our reliance on volatile witnesses. Consider the replicated object represented on figure 2. 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*. Assume now the following scenario:

(1) server *C* fails;

(2) server *A* fails before any other update occurs;

(3) server *C* recovers and marks its copy available since its version number matches that of the volatile witness;

(4) server *C* and witness *w* record several updates;

(5) server *C* and site *B* both fail;

(6) server *A* recovers.

If copy r_1 had kept its was-available set updated in real time, it would notice at recovery time that it was the last available copy of *X* and would not wait for the recovery of r_2 to mark itself available again. As a result, *X* would be left in an inconsistent state. The only possible way to avoid this type of occurrence would be to require witnesses to maintain their own was-available sets and to be included in the was-available sets of all copies. As a result, witnesses would need to have a part of their state stored in stable storage.

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 copies can only be excluded from an available set because they did not participate to an update, excluded sites will always be out of date and unable to recover by consulting a volatile witness.

4. DISCUSSION

A major advantage of our protocols 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 analyses of the protocol under standard Markov assumptions has indeed confirmed its excellent performance. protocol. There are however two important issues that need to be addressed. First, the correctness of the protocol depends on the fact that every operational witness receives all update requests. To ensure that, we need to locate witnesses on sites that are not overloaded and have sufficient time-out delays in the protocol used to broadcast update requests to all available sites. Second, the sites holding the witnesses are subject to failures. One possible solution would be to replace failed witnesses instead of waiting for the recovery of the failed site. This technique, known as *regeneration* [10], approximates the protection provided by additional witnesses, but at a much lower cost.

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. Barbara, H. Garcia-Molina, and A. Spauster, ‘‘Increasing Availability Under Mutual Exclusion Constraints with Dynamic Vote Reassignment,’’ *ACM TOCS*, 7, 4 (1989), pp. 394-426.

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

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

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

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

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

[9] J.-F. Pâris, ‘‘Voting with Bystanders,’’ *Proc. 9th ICDCS*, (1989), pp. 394-401.

[10] 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.

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