3-29-1990

A Bound of Data Availability when Networks Partition

Michael Goldweber
Dartmouth College

Donald B. Johnson
Dartmouth College

A BOUND ON DATA AVAILABILTY WHEN NETWORKS PARTITION

Michael Goldweber and Donald B. Johnson

Technical Report PCS-TR90-145

A Bound on Data Availability when Networks Partition

Michael Goldweber* Donald B. Johnson†
Dartmouth College‡

March 29, 1990

Abstract

Many consistency or replication control schemes that increase data availability in distributed systems exist, and the search for improvements continues, though there have been no good nontrivial upper bound demonstrating how much improvement is possible. We present a new upper bound for data availability under replication for general networks. In addition we also describe a new technique that yields near optimal levels of data availability with respect to this bound.

1 Introduction

In a distributed system subject to component failure, replication is typically used to improve data availability. A single failure in a non-replicated environment can cause a data item to become inaccessible. By storing copies of data on more than one processor, the probability that at least one copy is accessible increases.

The full benefits of data replication are in practice very difficult to achieve however because of the need for data *correctness*. By correctness

*e-mail address:michael.goldweber@dartmouth.edu
†e-mail address:djohnson@dartmouth.edu
‡Department of Mathematics and Computer Science. Hanover, N.H. 03755

we mean that the concurrent execution of transactions on replicated objects is equivalent to some serial execution on non-replicated data; a constraint known as *one-copy serializability*[10]. While in a non-replicated environment a local concurrency controller is sufficient to insure serializability, replicated systems also require a consistency control protocol to guarantee that all copies be mutually consistent (one-copy behavior). The problem of insuring mutual consistency is most interesting when site and link failures can partition the system into disjoint sets of communicating sites.

Partition resilient consistency control protocols that provide mutual consistency can be classified as either syntactic or semantic and as either optimistic or pessimistic[12]. These distinctions categorize protocols by how they define correctness and by when they detect and take action regarding a non-serializable operation. In this paper we examine the most general purpose class of protocols, pessimistic syntactic, where committed transactions are never backed out and the one-copy serializability correctness criterion is tested by examination of a transaction's read and writesets.

While the literature contains a large number of pessimistic syntactic protocols as well as a few studies comparing some of the more popular approaches, an understanding of the limitations of data replication on availability has remained elusive (see [12] for a good though slightly out of date survey of protocols and [20, 30, 32] for a sampling of the comparative studies). It has been shown that replication can provide superior availability over a non-replicated scheme. In addition it is also known that a transaction's probability of success must be less than the probability the issuing site is operational[20].

In this paper we demonstrate a new upper bound on data availability under replication for general networks. Use of this bound in concrete cases generally shows that data replication will not yield a significant increase in availability. The bound also gives insight in how to provide a high level of data availability. Accordingly we also present a new mechanism, the Segment Server, which achieves a nearly optimal level of data availability. The server utilizes a new protocol which is a variant of the Available Copies[9] protocol modified to guarantee serializable execution in partitionable networks.

The organization of this paper is as follows. In the next section we present the system model. In Section 3 we present our method for computing data availability. Sections 4 and 5 contain brief descriptions of some

previous protocols and upper bounds respectively. In Section 6 we present our new upper bound on data availability. Section 7 contains a description of our Segment Server. Finally we conclude the paper in Section 8 with a discussion of our results.

2 System Model

The environment we consider is composed of sites, bi-directional links and local network segments (ethernet of token ring). Links fail by crashing or by delaying or failing to transmit messages; partial failures such as links operating in only one direction or garbling messages are not considered. Our assumption is that any message transmitted is correct in its entirety and that the sequential order of transmitted messages is preserved. Local network segments, on the other hand never fail, but the processors (compute nodes, gateways and repeaters) connected to them are subject to failure. These segments also have the *non-partitioning property*, which insures that any two operating processors on the same segment will always be able to communicate. Processors and links are fail-stop[40]; although they may fail to send, receive, or transmit a message, byzantine failures[28] are not considered. Finally, all node and link failures are eventually repaired, although once repaired they are again subject to failure.

Since message passing is the only inter-node communication mechanism, processor and link failures can partition the network into disjoint sets of communicating sites. When communication between two processors is lost it is typically impossible for one site to determine whether the other has failed or is partitioned away. It is feasible though for one site to utilize the non-partitioning property of local network segments and knowledge of the initial topology to sometimes distinguish between a site failure and a partitioning.

We model the system as a distributed database, accessed through atomic transactions[21]. For ease of discussion our database is composed of a single data item, though our results are not dependent upon this. We assume that all transactions are well formed and that one transaction executed alone transforms the database from an initially correct state to another correct state[48]. Transactions can read or write the object. A read operation only conflicts with concurrently executing writes, while a write conflicts

with both concurrently executing reads and writes. Finally we assume the presence of a commit protocol which guarantees the atomic[43, 42] property for transactions.

3 Measures of Data Availability

One needs to first settle on a definition for data availability before a bound can be placed upon it. There are two definitions of availability in the literature[23]. The first is the probability at time t, as t goes to infinity, that an arbitrary site can legally and physically gain access to the data. The second is the probability at time t, as t goes to infinity, that there exists some site which can legally and physically gain access to the data.

We choose to present our results using the former definition rather than the more common latter one since it more accurately reflects a system user's viewpoint. A drawback of the second metric is that it fails to capture the influence of partition size on the probability a particular transaction at a particular site will succeed, thus favoring certain classes of protocols over others. Also for the second definition to be useful, transactions would require the knowledge of where to be submitted and have the ability of being submitted at that chosen site. We believe both of these assumptions are unrealistic.

Finally, in [20] it was shown that using the second metric, one can measure a level of data availability greater than the reliability of the underlying network. This study demonstrated how a transaction submitted to a site in a network with nodes of operational probability p had a success probability q, $q \geq p$, when q was measured using the more common second definition. Clearly it is uninstructive to use a metric which can allow transactions to have a greater chance of succeeding than the probability the submission site is operational.

4 Previous Protocols

Within the pessimistic syntactic protocol domain, there are two broad categories of approaches, static and dynamic. A static scheme is one that determines the current access partition solely by the site membership in

the different physical partitions[45], where an access partition is a set of communicating sites (a physical partition) containing at least one copy of the object and in which access to the object is allowed.[1] A property of all pessimistic schemes is that there is at most one access partition in the network at any given time (mutual exclusion), and two successive access partitions always overlap by at least one copy bearing site.

The most well known static protocol is Quorum Consensus[19], also known as static voting.[2] Each copy is allocated a number of votes and each data operation is assigned a threshold (or quorum) value. Transactions must acquire the appropriate threshold of votes from the copies it can communicate with, before executing the desired operation. The quorum values are set so that conflicting non-serializable sequences never occur. Majority Consensus[46] is a special case of Quorum Consensus since it specifies all operations to have their threshold be $\lceil (|n|+1)/2 \rceil$, n being the number of copies in the network.

Voting schemes are popular because they can be fine-tuned to reflect a desired availability by varying the initial vote assignment[17, 5] and operation thresholds. The ability to implement the Primary Copy Protocol[3, 44] by assigning all the votes to a single copy, and the capacity to favor reads over writes by choosing quorum values which allow simultaneous reads in multiple physical partitions by disabling writes in all partitions are examples of static voting's flexibility. Voting has been subject to more analysis than other protocols. This has included the determination of optimal vote assignments with respect to availability performance[2, 6, 26, 47], and of communication costs[27].

There are also some noteworthy variations on the basic voting protocol. Missing Writes[14] is a hybrid protocol that degenerates into Majority Consensus whenever the system is partitioned. Viewstamped replication[34, 35] improves upon some of Majority Consensus' inefficiencies and provides a front-end that appears like the simple Primary Copy Protocol to its users. Finally in [1] a method is presented that reduces the storage needs for a Quorum Consensus based protocol.

[1] An access partition is sometimes referred to as a majority partition. This is usually when a protocol requires that the current access partition contain a strict majority of sites from some previously defined set of sites.

[2] To facilitate our presentation, we capitalize the names commonly applied to the protocols we mention.

Not all static protocols are based upon vote assignments. Garcia-Molina and Barbara have shown that weighted voting cannot represent all of the valid access groupings that preserve mutual exclusion[4, 18], and therefore proposed coteries which are sets of site groupings that may perform an operation. Their Coterie Protocol is an access only protocol, which means that each data operation is assumed to conflict with all other operations, including itself. In terms of our database model this is the same as making no distinction between read and write operations; all accesses are considered to be writes. The protocol, as originally described, is inefficient in that it can require an exponential number of set operations before determining if an access can be allowed. Acceptance Sets[45], also based on coteries, utilizes a more efficient implementation.

Dynamic protocols typically adjust the parameters for determining the current access partition as component failure and recoveries alter the network topology. The most popular scheme is to have access partition $i+1$ be the physical partition that contains the majority of copies from access partition i. Version Vectors[13] was the first protocol of this type. Unfortunately it is based upon some unrealizable operational assumptions. Each site contains a symmetric and transitive P-bit connection vector indicating with which of the P sites it can communicate. The protocol requires that changes in the system configuration from site/link failures and recoveries be instantaneously recorded in the proper connection vectors.[3]

The Dynamic Voting[23] and Optimistic Dynamic Voting[37] protocols solved these implementation problems while providing the same level of data availability as Version Vectors. Various techniques have been developed to further improve these two protocols. Lexicographic or linear ordering[22, 24] provides a mechanism for tie-breaking. Topological Dynamic Voting[37] allows a site to utilize the non-partitioning property of local network segments and the initial network topology to sometimes distinguish between a site failure and a network partition. By storing the object's state information on additional sites (Witnesses), one can effectively increase the level of replication without incurring the associated storage overhead[36]. Finally a non-partitioning resilient protocol called

[3]Furthermore the recovery/merge algorithm as described does not guarantee mutual exclusion (leading to non-serializable executions). When a site/block merges with the majority it should adopt the majority's complete version vector (with the appropriate new zero entries) as well as the current value of the data object.

Regeneration[39] has been adapted to work with both static and dynamic voting protocols[31, 33].

Not all pessimistic dynamic schemes are based upon the majority of current copies. Class Conflict Analysis[41, 50] determines the set of transactions executable in each physical partition after every component failure. This mapping, though, requires an *a priori* characterization of all transactions. The Vote Reassignment class of protocols[7, 8] dynamically reassign votes upon site and link failures. Various techniques are presented on how to redistribute votes after a failure and what action to take when a failed copy recovers.

Hybrid protocols which merge both static and dynamic features have also been developed. Accessibility Thresholds[15, 16] is based on the concept of views. A site maintains an approximation of the set of sites it can communicate with, called a view. Each view then sets the quorum thresholds for itself in such a way as to prevent non-serializable executions both within the view and across all other views. Finally, the Dynamic Voting protocol has been adapted to revert to static voting whenever the size of the current access partition falls below a preset threshold[25].

5 Previous Upper Bounds

While it is desirable to achieve 100% data availability, it is clear from the above discussion that availability is limited from above by the average probability a transaction issuing site is operational. This observation, while useful in choosing between availability measures gives little insight into the performance of existing protocols relative to their potential.

The Oracle protocol[20] improved this bound through simulation, allowing for the first time a realistic understanding of replication's potential. Given a network topology, the individual component reliability characteristics, a distribution of copies, location and frequency distributions for transactions, and a sequence of events, the protocol determines the potential availability possible for the specified environment. By constructing a graph from the event stream and locating the maximum cost path, the protocol correctly outputs the assignment of access partitions maximizing data availability. The Oracle was used to demonstrate that when all transactions perform updates on a single replicated object, certain environments

already possess protocols that live up to replication's potential, while others do not[20].

6 A New Upper Bound

We alter our database model to be access only (i.e. no distinction is made between read and write operations with respect to conflicts and serializability). While this seems like a severe restriction, all of the above described protocols except Quorum Consensus and its direct variations exhibit this characteristic. Quorum Consensus also behaves in this manner whenever the quorums for reading and writing are set to the same value, which is typically done in order to maximize writer throughput.

A transaction mix is sometimes described in terms of the read rate α. There are α reads for every update to the object. The access only assumption is equivalent to setting $\alpha = 0$. We observe that when α is sufficiently high ($\alpha \geq |N|$, N being the set of sites in the network), full replication with a read one/write all protocol will yield a level of availability almost equal to the average probability a transaction issuing site is operational.

We model the network as a graph $G = (V, E)$ composed of a set V of sites connected by a set E of links. Each of the $|V \cup E|$ component's reliability is characterized by the percentage of time it is operational. Finally we designate $V' \subseteq V$ to indicate those sites that can issue a transaction against the object.

We observe that the data availability in a non-replicated environment, whose single copy has a reliability of p, can be improved by a factor of $1/p$, by increasing the reliability of the copy to 100%. We state this observation as the following lemma.

Lemma 1: *If x is the overall object availability in a non-replicated system whose single copy has a reliability of p, then exchanging that site with an infallible node will yield an availability of $(1/p)x$.*

A useful method of analyzing replication protocols is to measure the size of each access partition in terms of the number sites from V' it contains. This site count can be weighted to reflect any given transaction frequency distribution. We observe that protocols which maximize this quantity over

time will outperform those that do not.

Lemma 2: *If p is the probability that sites s and t can communicate and q is the probability that sites s, t, and the set U of sites can all communicate, where $s, t \notin U$ and $|U| \geq 1$, then $q \leq p$.*

Proof: p can be determined by enumerating the set of all $2^{|V \cup E|}$ network states and summing the occurence probabilities of each state that allow s and t to communicate. Only a subset of these states allow s, t and the other sites in U to communicate; therefore $q \leq p$. $\diamond$

There is at least one site i_{opt} in G that is optimally located so as to maximize the probability that it can communicate with V'. Let $Rel_{i,j}$ be the probability that site i, which is assumed to be operational, can communicate with j. We define the function

$$AvgSize_i = \sum_{j \in V'} Rel_{i,j} \tag{1}$$

to measure the expected number of sites from V' that are contained in i's physical partition. i_{opt} is a site where

$$AvgSize_{i_{opt}} = max_{i \in N} AvgSize_i$$

Furthermore the individual $Rel_{i,j}$'s can also be weighted to reflect any given transaction frequency distribution.

Theorem 1: *No replica control protocol using a set C of copies, $C \subseteq V$, can provide superior object availability than a single copy stored on an infallible node located at i_{opt}.*

Proof: Any access partition j that can form under a replica control protocol must contain $M_j \subseteq C$ copies. When $|M_j| = 1$ it is clear that $AvgSize_{M_j} \leq AvgSize_{i_{opt}}$. One concludes that the infallible copy at i_{opt} is better situated with respect to the set V' than a replicated copy in partition j. When $|M_j| > 1$ there is no individual site $k \in M_j$ such that $AvgSize_k > AvgSize_{i_{opt}}$. Furthermore by Lemma 2, for each $l \in V'$, the probability that l can communicate with all of the copies in M_j is not greater than the probability that l can communicate with just k. This indicates that there is a greater expected number of sites from V' capable of

communicating with just i_{opt} than with any $M_j \subseteq C$ copies.

Theorem 2: *If x is the overall object availability in a non-replicated system whose single copy located at i_{opt} has a reliability of p, then the upper bound on data availability with respect to replication is $(1/p)x$.*

Proof: Directly from Lemma 1 and Theorem 1. $\diamond$

Given that current technology can provide processors in the 95+% reliability range, effort should be focused on techniques which increase server reliability and optimize server placement, instead of on general replication schemes.

7 The Segment Server

The two keys to achieving our bound is locating the i_{opt} site and producing an infallible data server. The location problem can be reduced to determining the two site communication probability $Rel_{i,j}$. Unfortunately $Rel_{i,j}$ is $\#P$-Complete for general graphs[49, 38, 11]. There are some classes of graphs, though, for which polynomial algorithms exist. These include many typical configurations found in practice such as complete, series-parallel and other subclasses of planar graphs[11].

We now describe an architecture, which we call the Segment Server (SS), that produces a data service whose reliability can be made asymptotically close to one. The SS is physically composed of a single non-partitionable local network segment (ethernet or token ring) supporting a broadcast mechanism, with m processors connected to it. $n \subseteq m$ processors will both store a copy of the object and execute a special optimal consistency protocol, which we call the Segment Server Protocol (SSP). These sites which can be either current, failed or *comatose* also store a set of site ids called the *was-available* set. $l \subseteq m$ will act as gateways between the server segment and the remainder of the network. A node can store a copy, act as a gateway, perform both functions, or neither.

SSP is a variation of Long and Paris' improved version of the Available Copies protocol (AC)[9, 29]. The differences are that in SSP transactions originate from outside the non-partitionable segment and the local concurrency control (CC) functions are centralized, while in AC all transactions

originate from within the non-partitioning segment and the CC mechanism is fully replicated.

The n sites executing SSP designate one copy to act as the current primary whose responsibility it is to respond to all access requests and to maintain the CC mechanisms. Failure of the primary triggers the election of a new primary from among the set of operating copy-bearing nodes. A newly chosen primary's first task is to broadcast an abort token for all transactions in progress to the other $n-1$ copies. The protocol requires there be at most one primary at any given time to insure correct behavior.

An operation request originates at some site and is routed to any reachable operating gateway, which need not be kept the same for the duration of the transaction.[4] The gateway passes the request to the current primary using the broadcast facility. Read requests allowed by the CC function are satisfied by the primary who transmits the response back to the requester via any gateway. Write commands which clear the CC level are numbered, executed and ACKnowledged by the primary. All non-primary copies also receive the update information and the primary's ACK since both of these messages are transmitted using the broadcast facility. If the message receiving gateway is also the current primary, the command and its ACK are still broadcast in order to keep the other copies abreast of the updates.

When the primary receives the commit token it broadcasts on the segment, the CC mechanism's abort or commit decision regarding the transaction. The abort choice causes all copies to discard that transaction's updates and for the primary to relay this outcome to the requester. The primary, in the commit case, as part of the general transaction commit protocol broadcasts both the transaction's ID and its update count. Current copies that have received the full set of updates inform the primary of such, while the remainder of the operational copies become comatose. Finally the primary broadcasts the set of current site_ids (i.e. the set of current sites which received a complete set of updates) which becomes the was-available set for those sites.

A copy recovering from a failure is also considered comatose. A comatose copy is made current by locating any current copy and overwriting its data with the object's current state. Comatose copies, which are pro-

[4]Allowing the gateway to change provides the transaction a level of gateway fault tolerance.

hibited from being elected primary, remain comatose until a current copy is located.

A Segment Server will remain in continuous operation as long as a single current copy is operational. Recovery from a total failure in SSP is identical to the AC recovery scheme where the transitive closure of the was-available sets is used to indicate which host received the last update. Once the critical site is again operational other comatose copies can recover from it. A more complete description of the SS, approach including the recovery algorithm and the proof of the protocol's correctness can be found in the first author's forthcoming PhD thesis.

SSP's main shortcoming is that transactions must restart in the event of a primary's failure, which is due to the centralization of the CC function. A replicated CC mechanism, like any other replicated data, requires one-copy behavior for correctness. The AC protocol achieves a correct replicated CC function by assuming the existence of an *atomic* broadcast mechanism. Unfortunately using current technology, differences in work load and input queue length make atomic broadcast very difficult and expensive to implement. SSP accomplishes the same goals as AC without requiring its broadcast primitive to operate atomically.

The AC paradigm was chosen for the SS because AC is optimal with respect to availability for non-partitionable environments.[5] By increasing the number of copies SSP manages, the reliability of the server can be made arbitrarily close to perfect, therefore a SS represents a good approximation of an infallible server. While it is not known how to analytically analyze data availability in a network subject to partitioning, it has been shown in the first author's forthcoming PhD thesis, that through simulation, an optimally placed SS yields near optimal levels of data availability.

8 Conclusions

We have shown an upper bound on data availability under replication for a general class of protocols in an access only environment. This bound demonstrates that techniques which increase server reliability and improve server placement are more powerful in general than any standard replication

[5]The proof that AC is optimal in a non-partitionable system can be found in the first author's forthcoming PhD thesis.

scheme. It is better to approximate a single well placed infallible server than to replicate the object, creating a distributed server organized under a replica control protocol.

The Segment Server mechanism is presented as a means to construct a single highly reliable data server for networks subject to partitioning from component failures. By increasing the number of copies stored on the segment, the overall reliability of the server can be made arbitrarily close to that of an infallible site. Finally, the Segment Server was shown to yield a level of availability very close to optimal.

References

[1] Divyakant Agrawal and Amr El Abbadi. Reducing storage for quorum consensus algorithms. In *Proceedings of the 14th International Conference on Very Large Data Bases*, pages 419–430, 1988.

[2] Mustaque Ahamad and Mostafa H. Ammar. Performance characterization of quorum consensus algorithms for replicated data. In *Proceedings of the 6th Symposium on Reliability in Distributed Software and Database Systems*, pages 161–168. IEEE, 1987.

[3] P. A. Alsberg and J. D. Day. A princple for resilient sharing of distributed resources. In *Proceedings of the 2nd Annual Conference on Software Engineering*, pages 627–644, October 1976.

[4] Daniel Barbara and Hector Garcia-Molina. Mutual exclusion in partitioned distributed systems. *Distributed Computing*, 1:119–132, 1986.

[5] Daniel Barbara and Hector Garcia-Molina. The vulnerability of vote assignments. *ACM Transactions on Computer Systems*, 4(3):187–213, August 1986.

[6] Daniel Barbara and Hector Garcia-Molina. The reliability of voting mechanisms. *IEEE Transactions on Computers*, C–36(10):1197–1208, 1987.

[7] Daniel Barbara, Hector Garcia-Molina, and Annemarie Spauster. Policies for dynamic vote reassignment. In *Proceedings of the 6th International Conference on Distributed Computing Systems*, pages 37–44, May 1986.

[8] Daniel Barbara, Hector Garcia-Molina, and Annemarie Spauster. Protocols for dynamic vote reassignment. In *Proceedings of the 5th ACM Symposium on Principles of Distributed Computing*, pages 195–205. ACM, 1986.

[9] P. A. Bernstein and N. Goodman. An algorithm for concurrency control and recovery in replicated distributed databases. *ACM Transactions on Database Systems*, 9(4):596–615, December 1984.

[10] Philip A. Bernstein, Vassos Hadzilacos, and Nathan Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley, 1987.

[11] Charles J. Colbourn. *The Combinatorics of Network Reliability.* Oxford University Press, 1987.

[12] Susan B. Davidson, Hector Garcia-Molina, and Dale Skeen. Consistency in partitioned networks. *ACM Computing Surveys*, 17(3):341–370, September 1985.

[13] Dančo Davčev and Walter A. Burkhard. Consistency and recovery control for replicated files. In *Proceedings of the 10th ACM Symposium on Operating Systems Principles*, pages 87–96, December 1985.

[14] Derek L. Eager and Kenneth C. Sevcik. Achieving robustness in distributed database systems. *ACM Transactions on Database Systems*, 8(3):354–381, September 1983.

[15] Amr El Abbadi. *A Paradigm for Concurrency Control Protocols for Distributed Databases.* PhD thesis, Cornell University, August 1987.

[16] Amr El Abbadi and Sam Toueg. Availability in partitioned replicated databases. In *Proceedings of the 5th ACM Symposium on Principles of Database Systems*, pages 240–251, March 1986.

[17] Hector Garcia-Molina and Daniel Barbara. Optimizing the reliabililty provided by voting mechanisms. In *Proceedings of the 4th International Conference on Distributed Computing Systems*, pages 340–346, October 1984.

[18] Hector Garcia-Molina and Daniel Barbara. How to assign votes in a distributed system. *Journal of the ACM*, 32(4):841–860, October 1985.

[19] D. K. Gifford. Weighted voting for replicated data. In *Proceedings 7th ACM SIGOPS Symposium on Operating Systems Principles*, pages 150–159, Pacific Grove, CA, December 1979.

[20] Michael Goldweber, Donald B. Johnson, and Larry Raab. A comparison of consistency control protocols. Technical Report PCS-TR89-141, Dartmouth College, 1989.

[21] J. Gray. Operating systems: An advanced course. In R. Bayer, R. M. Graham, and G. Seegmuller, editors, *Lecture Notes in Computer Science Vol. 60*. Springer Verlag, 1979.

[22] Sushil Jajodia. Managing replicated files in partitioned distributed database systems. In *Proceedings of the 3rd International Conference on Data Engineering*, pages 412–418. IEEE, February 1987.

[23] Sushil Jajodia and David Mutchler. Dynamic voting. In *Proceedings of the 1987 SIGMOD International Conference on Management of Data*, pages 227–237, 1987.

[24] Sushil Jajodia and David Mutchler. Enhancements to the voting algorithm. In *Proceedings of the 13th International Conference on Very Large Data Bases*, pages 399–406, 1987.

[25] Sushil Jajodia and David Mutchler. Integrating static and dynamic voting protocols to enhance file availability. In *Proceedings of the 4th International Conference on Data Engineering*, pages 144–153. IEEE, February 1988.

[26] Akhil Kumar and Arie Segev. Opimizing voting-type algorithms for replicated data. In *Advances in Database Technology EDBT '88*, pages 428–442. Springer–Verlag, 1988. In: Lecture Notes in Computer Science Vol. 303.

[27] Akhil Kumar and Arie Segev. Optimizing and evaluating algorithms for replicated data concurrency control. In *Proceedings of the 9th International Conference on Distributed Computing Systems*, pages 101–109, June 1989.

[28] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages*, 4(3):382–401, July 1982.

[29] Darrel D. E. Long and Jehan-Francois Paris. On improving the availability of replicated files. In *Proceedings of the 6th Symposium on Reliability in Distributed Software and Database Systems*, pages 77–83. IEEE, 1987.

[30] Darrel D. E. Long and Jehan-Francois Paris. A realistic evaluation of optimistic dynamic voting. In *Proceedings of the 7th Symposium on Reliable Distributed Systems*, pages 129–137. IEEE, October 1988.

[31] Darrel D. E. Long and Jehan-Francois Paris. Regeneration protocols for replicated objects. In *Proceedings of the 5th International Conference on Data Engineering*, pages 538–545. IEEE, 1989.

[32] Darrel D. E. Long, Jehan-Francois Paris, and John L. Carroll. Reliability of replicated data objects. In *8th Annual International Phoenix Conference on Computers and Communications*, pages 402–406. IEEE, March 1989.

[33] Darrell D. E. Long, John Carroll, and Kris Stewart. The reliability of regeration-based replica control protocols. In *Proceedings of the 9th International Conference on Distributed Computing Systems*, pages 465–473, June 1989.

[34] Brian M. Oki. Viewstamped replication for highly available distributed systems. Technical Report MIT/LCS/TR-423, Massachusetts Institute of Technology, August 1988.

[35] Brian M. Oki and Barbara Liskov. Viewstamped replication: A new primary copy method to support highly-available distributed systems. In *Proceedings of the 7th Annual ACM Symposium on Principles of Distributed Computing*, pages 8–16. ACM, 1988.

[36] Jehan-Francois Paris. Voting with witnesses: A consistency scheme for replicated files. In *Proceedings of the 6th International Conference on Distributed Computing Systems*, pages 606–612, May 1986.

[37] Jehan-Francois Paris and Darrel D. E. Long. Efficient dynamic voting algorithms. In *Proceedings of the 4th International Conference on Data Engineering*, pages 268–275. IEEE, February 1988.

[38] J. S. Provan and M. O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, (12):777–788, 1983.

[39] C. Pu, Jerre D. Noe, and A. Proudfoot. Regeneration of replicated objects: A technique and its Eden implementation. *IEEE Transactions on Software Engineering*, SE-14(7):936–945, July 1988.

[40] R. D. Schlichting and F. B. Schneider. Fail stop processors: An approach to designing fault-tolerant computing systems. *ACM Transactions on Computer Systems*, pages 222–238, 1983.

[41] D. Skeen and D. Wright. Increasing availability in partitioned networks. In *Proceedings of the 3rd ACM Symposium on Principles of Database Systems*, pages 290–299, April 1984.

[42] Dale Skeen. *Crash Recovery in a Distributed Database System.* PhD thesis, University of California, Berkeley, May 1982. ERL Memo M82/45.

[43] Dale Skeen and Michael Stonebraker. A formal model of crash recovery in a distributed system. In *Proceedings 5th Berkeley Workshop on Distributed Data Management and Computer Networks*, pages 129–142. ACM/IEEE, 1981.

[44] Michael Stonebraker. Concurrency control and consistency of multiple copies of data in distributed INGRES. *IEEE Transactions on Software Engineering*, SE-5(3):188–194, May 1979.

[45] Jian Tang and N. Natarajan. A static pessimistic scheme for handling replicated databases. In *Proceedings of the 1989 SIGMOD International Conference on Management of Data*, pages 389–398, 1989.

[46] R. Thomas. A majority consensus approach to concurrency control. *ACM Transactions on Database Systems*, 4(2):180–209, June 1979.

[47] Z. Tong and R. Y. Kain. Vote assingments in weighted voting mechanisms. In *Proceedings of the 7th Symposium on Reliable Distributed Systems*, pages 138–143. IEEE, October 1988.

[48] I. Traiger, J. Gray, C. Galeieri, and B. Lindsay. Transactions and consistency in distributed database systems. *ACM Transactions on Database Systems*, 7(3):323–342, September 1982.

[49] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, (8):410–421, 1979.

[50] D. D. Wright. Managing distributed databases in partitioned networks. Technical Report 83-572, Cornell University, September 1983.