11-11-1991

Availability Issues in Data Replication in Distributed Database

Donald B. Johnson
Dartmouth College

Larry Raab
Dartmouth College

AVAILABILITY ISSUES IN DATA REPLICATION IN DISTRIBUTED DATABASE

Donald B. Johnson
Larry Raab

Technical Report PCS-TR91-168

Availability Issues in Data Replication in Distributed Database

Donald B. Johnson* Larry Raab†
Dartmouth College‡

November 11, 1991

Abstract

Replication of data at more than one site in a distributed database has been reported to increase the availability of data in systems where sites and links are subject to failure. We have shown in results summarized in this paper that in many interesting cases the advantage is slight. A well-placed single copy is available to transactions almost as much of the time as is correct replicated data no matter how ingeniously it is managed. We explain these findings in terms of the behaviour of the partitions that form in networks where components fail. We also show that known and rather simple protocols for the maintenance of multiple copies are essentially best possible by comparing them against an unrealizable "protocol" that knows the future. We complete our study of these questions by reporting that while computing the availability of data is $\#P$-Complete, nonetheless there is a tight analytical bound on the amount replication can improve over a well-located single copy. We close with some observations regarding system design motivated by this work.

1 Introduction

Two distinct reasons to replicate data at more than one site in a distributed database are to increase availability and to reduce cost. In this paper we study the effects of replication on availability, which we define below as the probability that a request for access to a data item will succeed.

The issue we are interested in is not whether the database program will allow a request for the normal reasons of consistency and serializability of transactions, but how well the network will accommodate requests in the presence of component failures. It may be that, because of site or link failures, a copy of the data is not

*e-mail address:djohnson@dartmouth.edu
†e-mail address:raab@dartmouth.edu
‡Department of Mathematics and Computer Science, 6188 Bradley Hall, Hanover, N.H. 03755

up to date, or it may be that failures have isolated a requesting site from sites at which data reside.

This last phenomenon is called *network partitioning*, and it is what the network designer should attempt to minimize or ameliorate, and what makes the problem of great interest to the analyst. We approach the problem from the viewpoint of analysis, but we believe that the results of whatever analysis can be accomplished should be of interest and value to network and distributed database designers.

Opportunities for analysis and invention occur at several levels. At the lowest level, we can seek to characterize the probability that two sites in an arbitrary network will be connected, or that some larger subset of sites will be mutually connected. At a higher level, we can define protocols to maintain copy consistency and the appearance to requests for data that there is only one copy, and that this virtual copy (so to speak) is consistent with an entire history of accesses, which may be both reads and writes. This desired property is called *one-copy serializability* and, as we will recount shortly, one-copy serializability captures what is the usual notion of "correctness" in a distributed database.

There are, however, some formidable obstacles to certain aspects of this analysis. For example, computing the probability in an arbitrary network, with components subject to failure and recovery at rates described by given probability distributions, that two given sites can communicate is almost surely intractable (it is in fact #P-Complete[Val79] as we discuss shortly). As we will discuss, availability depends on the probability that communication is possible, so availability is also difficult to compute. Nonetheless, we will have some interesting bounds on availability to present.

At the higher level, it is not a trivial matter to devise protocols to maintain one-copy serializability while allowing as many accesses as possible to be satisfied. In fact, the development of this field is marked by a sequence of inventions of protocols, each with improved performance over those known before. One of our contributions to this field is results that show that known protocols can be improved on very little if at all under conditions we will explain in some detail later. Simply stated, known protocols are close to best possible when write accesses dominate read accesses to the database.

It must be said in this context that the control of replicated data is complex and adds overhead to a system. But, then, replication is introduced to increase the probability that data will be available. So, there should be some advantage to offset the costs of added overhead. Another of our results is that under certain plausible conditions this advantage does not in fact materialize because replication is not very much better than maintaining a single copy. Essentially, the result is that if you know where to best locate a single copy, no replication scheme and protocol will give much better availability than will be had with a single data object.

In this paper we will discuss these results and give references to our papers where they have first appeared. Then we will discuss the implications we see for

these results in the design of distributed databases both for existing networks and for the design of such systems from the network on up to the location of data and the protocols for its management.

1.1 Network and Database Model

As suggested above, we postulate a network composed of sites, where data elements are stored and access requests for items are submitted, and bidirectional links over which sites at their endpoints communicate.

Sites and links are subject to failure and subsequent recovery according to some assumed probability distributions. There is a considerable variation of distributions over which our results hold, but it suffices for the purposes of this paper to postulate an exponential distribution for failures and a Poisson process for recovery[CL89].

We also make a possibly less realistic assumption with regard to sites and links, namely that they are fail stop. In other words, the only mode of failure assumed to occur is complete failure. Without this assumption, protocols for correct operation become immensely more complex. Aside from the difficulties of working with possibly malicious failure modes, consideration of such problems obscures the issues we address, namely, strategies for managing data in a practical environment where components fail.

In general there will be copies of items at various sites, and there will be a stream of access requests at various, possibly other, sites. The problem of data and request management is to obtain the same responses to requests as would be obtained were all data stored at a single site and all requests were submitted at this same site and managed in a serializable manner, the usual notion of database correctness. As mentioned above, this property is called one-copy serializability. In a completely reliable system, obtaining one-copy serializability is not trivial, but for the purposes of this paper, we will assume that the problem of one-copy serializability in a reliable system is well understood and solved, as indeed it is[BHG87].

The complexities with which we deal in this paper arise because both sites and links are subject to failure and subsequent recovery, and in this process the network may become disconnected and reconnected again.

1.2 Correct Protocols

Because of the partitioning just alluded to, there must be some control protocol, distributed over sites with data, that maintains correctness of data. The main problem is that when the network partitions, accesses that change the data can only be allowed in one block of the partition, else data will become inconsistent. Furthermore, reads in some other block of the partition may read data that is no longer correct because it has been changed elsewhere.

We call the block of a partition in which accesses are allowed the *distinguished block*. Necessary rules for any correct protocol to enforce are (1) *uniqueness:* there

may be only one distinguished block at a time and (2) *overlap:* successive distinguished blocks must have one copy of any data item in common in order for the most up-to-date copy to be accessed.

1.3 Availability

We define the *availability* of a data item to be the expectation over all sites of the probability that an access will succeed when the system is controlled by some protocol. Thus availability depends on the network, the location of data in the network, and the distributions governing failures, recoveries, and access requests, as well as the protocol used. The objective we assume in this paper is the maximization of availability, given a network and the distributions just mentioned. Thus the factors which can be varied are the protocol and the location of data. We will generally, therefore, view availability as a function of the protocol and location of data, and secondarily as a function of network topology.

1.4 The Oracle Protocol

In previous work[JR91b], we postulated a most powerful correct protocol, one that chooses at each event in the network the distinguished block with full knowledge of the future, both with respect to access requests and failures and recoveries, subject only to satisfying the overlap property. We call this protocol (which of course we don't know how to implement on-line) the *oracle protocol.* Clearly, any availability obtained with the oracle is an upper bound for any correct protocol when the other factors are fixed.

1.5 Good Protocols

In this section we introduce a number of protocols that have been proposed to maximize availability while guaranteeing the consistency constraints. Unlike the oracle, these protocols can be implemented in real systems, since they do not make use of knowledge of future events. Although many interesting and effective protocols have been introduced[BGMS89, DB85, ET86, GMB85, Gif79, Her87, JM88], we describe only those which we employ to illustrate points in this paper.

The **primary copy** protocol[AD76] is a generalization of a non-replicated data object. Although more than one copy may exist in the network, there exists a particular copy, designated the "primary", with which all accesses must communicate. This scheme is the simplest and easiest to implement, and it is against this scheme that replication protocols ultimately compete. In some networks, including certain networks studied in this paper, determining the optimal location of the primary copy is easy. However, finding the optimal location is in general $\#P$-Complete[JR91a]. In section 4, we give an effective, practical method for approximating this location based upon the past performance of the system.

The **majority consensus** protocol[Tho79], *MC*, is an instance of the quorum consensus protocol[Gif79], where no distinction is made between read and write operations. In the majority consensus protocol, each copy is assigned a number called its *vote assignment*, and a block is distinguished if and only if the total votes of all the copies in the block sum to a majority of the votes in the network. Since a majority of the votes is always required, it is clear that the consistency constraints are fulfilled.

The majority consensus protocol can be fine-tuned to maximize availability by varying the voting assignment. The best vote assignment will depend upon the topology of the network, the reliability of the components, and the distribution of the data accesses. Since in general finding the best vote assignment is computationally intractable, heuristics have been suggested to help choose an initial vote assignment[BGM87, MW88]. For this analysis, we use a uniform vote assignment of one vote per copy, since in our experiments the data access distribution and component reliabilities are all uniform and the topologies are roughly symmetric. By allocating to one copy a majority of the votes, majority consensus can achieve the same accessibility as the primary copy protocol, but this case must be selected before system startup and would not exhibit any benefits of replication.

The **majority of current** protocol[JM87, JM90, PL88], *MOC*, is equivalent to an earlier protocol by Davčev and Burkhard[DB85] except that *MOC* does not require that each site obtain instantaneous knowledge of all changes in network status. Two versions, identical for our purposes in functionality but differing in implementation, were presented by Jajodia and Mutchler[JM87, JM90] and by Long and Pâris[PL88].

The majority of current protocol assigns votes to each copy as does the majority consensus protocol, but *MOC* defines the distinguished block as the block containing a majority of the votes from *current* copies. A copy is *current* if it was updated during the last access to that data object. Since a *majority* of the current copies must be present, at most one distinguished block may exist at a time. Since a distinguished block must contain *current* copies, successive distinguished blocks have at least one copy in common. Thus the consistency constraints are fulfilled.

2 Simulation Results

We now report on simulations which shed light on two important questions, (1) how good are known protocols under full replication? and (2) of how much value on the average is replication itself? Data which elucidates these questions were obtained for a range of network topologies, progressively more connected, over several protocols, and over varying degrees of replication.

The results we report were obtained by simulation of a network with 101 sites and a stream typically of $1,000,000$ accesses generated randomly according to a uniform distribution over sites and according to a Poisson process at each site. For

the results given, we assumed site and link reliability to be 96% and a ratio of mean time-to-next-access to mean time-to-next-failure of 1/128. The simulations were run on a DECstation 5000 Model 200 and took from one to two hours depending primarily on how many links were in the network. Simulations were run long enough to obtain results with 95% confidence[JR91b, JR91c].

2.1 Network Topologies

For these results, the network of 101 sites (numbered from 0 to 100) was connected as a ring which we call Topology 0. Topology 1 was the ring to which has been added a chord from site 0 to site 50. Topology i was the ring with i such chords are added essentially equally spaced around the ring. The number 101 is somewhat arbitrary. An odd number simplifies the protocols, and a number in the vicinity of 100 allowed simulations to run within the limits of our computational resources while being more that a factor of 10 greater than the sizes of networks on which simulations had hitherto been reported in the literature.

2.2 Performance of Protocols

We have evaluated the performance of known "good" protocols against the oracle in systems with data replicated at every site so as to take advantage of the presumed value of replication. Surprisingly to us, we found that known and rather simple protocols perform on the average almost as well as the oracle. Thus, in our opinion, there is little of practical value to be gained from improving protocols.

As can be seen from Figure 1, *MOC* performs within six percentage points of optimal for all topologies and performs optimally for topologies 16[1], 256, and 4949. As the ratio of the mean time-to-next-access to the mean time-to-next-failure decreases, the performance of *MOC* relative to the oracle continues to improve[JR91b]. We conclude that under this model there is little room for the development of a protocol with performance superior to that of *MOC*. The only exception may be improved performance in very low connectivity networks (i.e. the ring and topology 1), but the availability in these networks is so low that they would be undesirable even at optimal performance.

It is also interesting to note that even the simple protocol *MC* performs nearly optimally for highly connected topologies. Furthermore, it can be proven that the performance of *MC* approaches the oracle as the size of a highly-connected network increases[JR91d]. From Figure 1, it is clear that the network need not be very large for this phenomenon to be observed.

[1] It is interesting to note that Topology 16 has only 16% more edges than Topology 0.

2.3 Performance of Replication

Turning our attention to replication in general, we see that although the oracle and *MOC* perform the best, the primary copy protocol frequently does nearly as well. As Figure 1 indicates, the average performance advantage of *MOC* over the primary copy protocol is only 2.9 percentage points, and the maximum advantage is 5.9 percentage points. At this point of maximum difference, the primary copy protocol yields availability 59.1% and *MOC* yields availability 65.0%. This advantage decreases as the level of replication decreases. In many applications, the increases in performance due to replication may be too small to warrant the additional hardware and software complexity and communication costs incurred by replication. Ultimately this determination can only be made by the designers of a particular system.

3 Analytical Results

Quite surprisingly, in view of the difficulty of computing availability itself, we are able to present a tight bound on the value of replication. The result characterizes the degree to which any possible replication scheme and protocol can improve over the availability of a single copy, provided the single copy is well-located. The bound is tight in the sense that there are networks for which any smaller bound can be exceeded by a very simple protocol, *MC*.

Theorem 1 [JR91d] *Let some network with a fixed access distribution achieve availability A with a single, well-located copy. Then no replication scheme with any protocol that guarantees mutual exclusion can achieve availability on this same network greater than* $\sqrt{A}$.

We will not repeat the proof here. However, we observe that this bound is tight for all A, $0.25 \leq A \leq 1.0$. What this means is that while our simulations give evidence that on the average the performance of single copies is much closer to best performance in replicated systems, this cannot be shown in general analytically. For example, in section 2.3 we reported that the greatest difference in availability we found was 59.1% for single copy versus 65.0% for a best replicated system. The above theorem gives an upper bound of 76.9%, quite loose relative to what we achieved in the simulations we ran.

At higher single-copy availabilities this bound are more useful. For example, if a single copy were to give a best availability of 92%, then the best replication could achieve would be 96%. These numbers are exactly the numbers we found in our simulations.

4 Placement

In order to be assured that the performance of a single copy protocol is within a square of optimal, we need not find the "best" location (the location that maximizes availability) for the single copy. It is sufficient to find a site that we expect to be in a block of size greater than the average expected block size over all sites. Unfortunately, finding off-line either the best site or a site with block size larger than average is a $\#P$-complete problem[JR91a]. Further, finding such a site on-line is impossible, since this would require that a site determine the likelyhood of request submissions in other blocks, which in turn would require communication between blocks. Fortunately, in most systems we can efficiently find a site yielding single copy performance at least as good these sites.

Although $\#P$-complete in general, the placement problem, as we refer to the determination of the optimal location for the data item, is solvable for some systems. Since often a network for an existing database is built incrementally around the database, the current location may be optimal. In addition, the single copy availability can be efficiently determined for regular network topologies[BGM87, Gil59], such as ring, single-bus, fully-connected, and for series-parallel networks[Col87, SW85]. Since, for these topologies, the single copy availability can be calculated in polynomial time, the placement problem can be solved in polynomial time simply by calculating this availability for each site in the network.

Although calculating the single copy availability is feasible in some special cases, it is unnecessary and perhaps undesirable to do so in real systems. Instead, each site x can record the actual number of access requests submitted to the other sites with which x can communicate, and the site x with the largest such count can be made the location of the copy. If the past network performance and access distribution is, as one would expect, indicative of future behavior, then this technique leads to optimal copy placement. This method does not require a priori knowledge of the network topology, hardware reliability, or access distribution, and adjusts automatically to unanticipated changes in any of the these system parameters. These characteristics are precisely those necessary for an automated single copy migration protocol, as we discuss in section 6.1. Our experience with simulation[JR91b] indicates that this approach will be successful.

5 Read/Write Ratio

Thus far in this paper we have not differentiated between read and update requests, thereby subjecting read accesses to the same consistency constraints as update accesses. Although we will argue that this policy maximizes availability in many systems, this is clearly not always the case. For example, when all accesses are reads, then availability is maximized by placing one copy at each site.

Analyses by Ahamad and Ammar on networks not subject to partitioning proves

that requiring a majority of votes for both read and write accesses is optimal for a wide range of network parameters[AA87]. In addition, our own simulations on the large, partitionable networks described in section 2 show that either this policy or read-one, write-all maximizes availability for all networks tested[JR91c]. In particular, availability is maximized when reads and updates are treated the same for systems with highly-connected topologies or low read rates.

6 System Design

It is evident that a study only of availability in the context of our model leaves many other questions to be answered in the design of systems. Among these questions is the question of cost, which we have touched on only indirectly in our observation that if replica control is of only marginal value in enhancing availability then its costs are likely not justifiable. In this concluding section we make some observations about the accommodation of systems to changing localities of access requests by data migration, the maintenance of robust single copies, and a possible standpoint for planning to cope with catastrophes.

6.1 Data Migration

The results given thus far are applicable to a system with a fixed access request distribution. Clearly, if the access distribution is allowed to change, in turn causing the optimal location of a single copy to change, then the single copy must be allowed to move to the new optimal location. In [JR91d], we prove that the bound presented in section 3 still holds, provided that the single copy does not cross partition boundaries. This assumption is not unreasonable, since changes in network topology are seldom drastic and since failures are typically of short duration. Therefore, a single data item will perform nearly as well as a replicated data item in both a system that has a fixed access request distribution and in a system that allows the data item to migrate in response to a changing access request distribution.

6.2 Maintaining Single Copies

Copies of data are often made in systems for purposes of backup. This is a form of replication that we have not considered in this paper, because it is not relevant to the question of availability as we have defined it. It is clear that there is the need for backups and checkpoints on site for protection against irretrievable data loss from nonvolatile storage. Our model is that the protection given by backups on site is reflected in the assumptions made about time to recovery following failure with backup copies that are not on-line in the sense that the copies which we have discussed are.

On the other hand, it may be reasonable to make sites more reliable by building redundancy into hardware or into copies of data within a single site. This is an

interesting problem, but it is distinct from the problem this paper addresses.

6.3 Catastrophe Protection

While it is reasonable to model failure processes probabilistically, it may not be so reasonable to model recovery processes probabilistically. If we define a catastrophe as an event that incapacitates a whole site for a long period of time, then it is evident that recovery may take such a long time as to be unacceptable, no matter what the "averages" say. It makes sense to maintain one or more copies at some other site or sites for purposes of catastrophe recovery. The question is whether it is also reasonable to provide for automatic catastrophe recovery by the mechanisms of replication we have discussed in this paper or whether it is more reasonable to expect human or extraordinary intervention for restoration after catastrophe. We argue that the latter is almost certainly preferable on the basis that catastrophe is or ought to be so rare for there to be no justification for incurring the cost of running replication protocols with continual voting or whatever during long periods of normal operation when single copies will suffice.

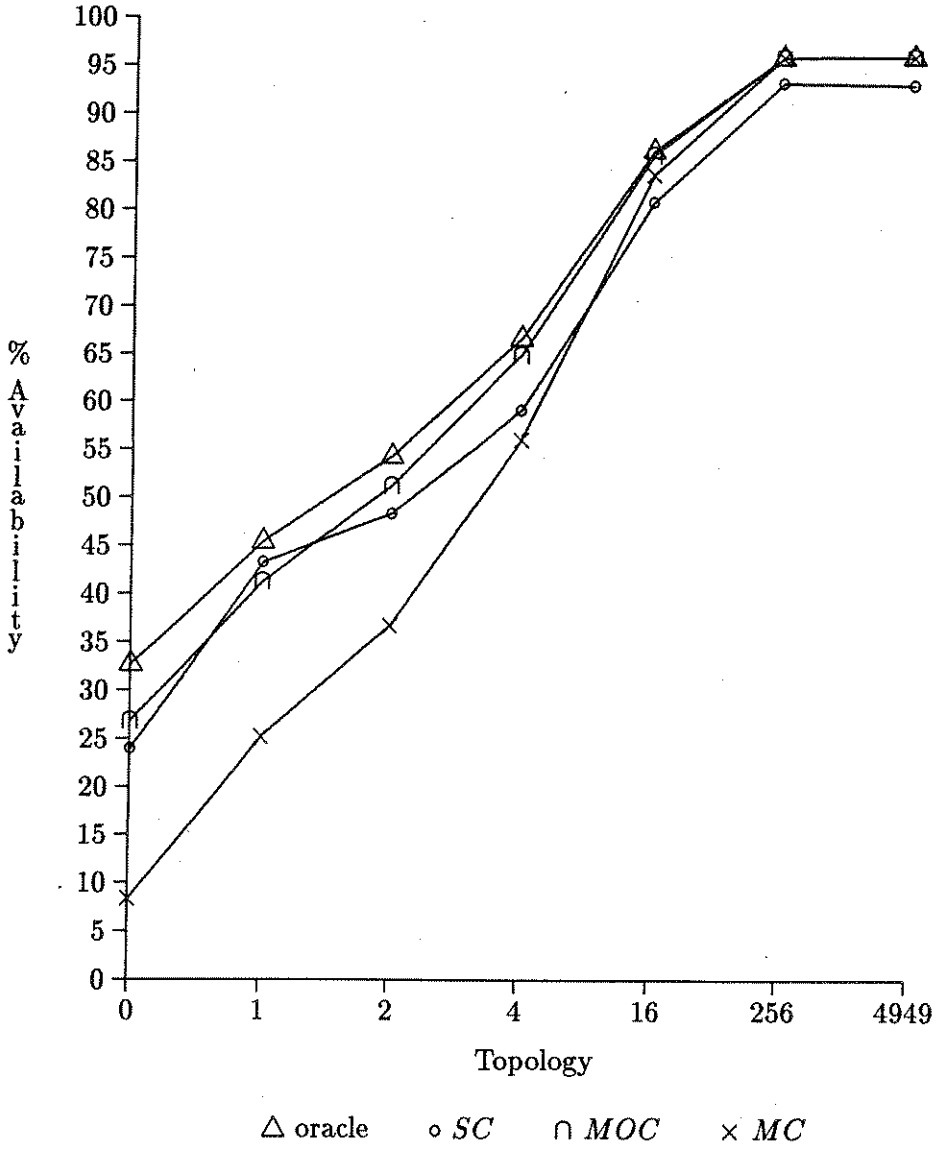

Figure 1: Protocol Comparison
(site & link reliabilities 96%)
(one copy per site)

References

[AA87] 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.

[AD76] P. A. Alsberg and J. D. Day. A principle for resilient sharing of distributed resources. In *Proceedings of the 2nd Annual Conference on Software Engineering*, pages 627–644, October 1976.

[BGM87] Daniel Barbara and Hector Garcia-Molina. The reliability of voting mechanisms. *IEEE Transactions on Computers*, C–36(10):1197–1208, 1987.

[BGMS89] Daniel Barbara, Hector Garcia-Molina, and Annemarie Spauster. Increasing availability under mutual exclusion constraints with dynamic vote reassignment. *ACM Transactions on Computer Systems*, 7(4):394–426, 1989.

[BHG87] Philip A. Bernstein, Vassos Hadzilacos, and Nathan Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley, 1987.

[CL89] John Carroll and Darrell D. E. Long. The effect of failure and repair distributions on consistency protocols for replicated data objects. In *Proceedings of the 22nd Annual Simulation Symposium*, pages 47–60. IEEE, 1989.

[Col87] Charles J. Colbourn. *The Combinatorics of Network Reliability.* Oxford University Press, 1987.

[DB85] 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.

[ET86] Amr El Abbadi and Sam Toueg. Availability in partitioned replicated databases. In *Proceedings of the 5th ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 240–251. ACM, March 1986.

[Gif79] 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.

[Gil59] E. N. Gilbert. Random graphs. *Annals of Mathematical Statistics*, 30:1141–1144, 1959.

[GM] Hector Garcia-Molina. Personal communication.

[GMB85] Hector Garcia-Molina and Daniel Barbara. How to assign votes in a distributed system. *Journal of the ACM*, 32(4):841–860, October 1985.

[Her87] Maurice Herlihy. Dynamic quorum adjustment for partitioned data. *ACM Transactions on Database Systems*, 12(2):170–194, June 1987.

[JM87] Sushil Jajodià and David Mutchler. Dynamic voting. In *Proceedings of the SIGMOD Annual Conference*, pages 227–237. ACM, May 1987.

[JM88] 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.

[JM90] Sushil Jajodia and David Mutchler. Dynamic voting algorithms for maintaining the consistency of a replicated database. *ACM Transactions on Database Systems*, 15(2):230–280, June 1990.

[JR91a] Donald B. Johnson and Larry Raab. Complexity of network reliability and optimal database placement problems. Technical Report PCS-TR91-178, Dartmouth College, June 1991.

[JR91b] Donald B. Johnson and Larry Raab. Effects of replication on data availability. *International Journal of Computer Simulation*, 1991. to appear.

[JR91c] Donald B. Johnson and Larry Raab. Finding optimal quorum assignments for distributed databases. In *Proceedings of the 1991 International Conference on Parallel Processing*, volume 3, pages 214–218. CRC Press, August 1991.

[JR91d] Donald B. Johnson and Larry Raab. A tight upper bound on the benefits of replication and consistency control protocols. In *Proceedings of the 10th Symposium on Principles of Database Systems*, pages 75–81. ACM, May 1991.

[MW88] Amir Milo and Ouri Wolfson. Placement of replicated items in distributed databases. In *Advances in Database Technology EDBT '88*, pages 415–427. Springer–Verlag, 1988. In: Lecture Notes on Computer Science Vol. 303.

[PL88] Jehan-François Pâris and Darrell D. E. Long. Efficient dynamic voting algorithms. In *Proceedings of the 4th International Conference on Data Engineering*, pages 268–275. IEEE, February 1988.

[SW85] A. Satyanarayana and R. K. Wood. A linear time algorithm for computing k-terminal reliability in series-parallel networks. *SIAM Journal of Computing*, 14:818–832, 1985.

[Tho79] R. Thomas. A majority consensus approach to concurrency control. *ACM Transactions on Database Systems*, 4(2):180–209, June 1979.

[Val79] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, August 1979.