11-14-1991

Effects of Replication on the Duration of Failure in Distributed Databases

Donald B. Johnson
Dartmouth College

Larry Raab
Dartmouth College

EFFECTS OF REPLICATION ON THE DURATION OF FAILURE IN DISTRIBUTED DATABASES

Donald B. Johnson
Larry Raab

Technical Report PCS-TR91-169

Effects of Replication on the Duration of Failure in Distributed Databases

Donald B. Johnson* Larry Raab†
Dartmouth College‡

November 14, 1991

Abstract

Replicating data objects has been suggested as a means of increasing the performance of a distributed database system in a network subject to link and site failures. Since a network may partition as a consequence of such failures, a data object may become unavailable from a given site for some period of time. In this paper we study duration of failure, which we define as the length of time, once the object becomes unavailable from a particular site, that the object remains unavailable. We show that, for networks composed of highly-reliable components, replication does not substantially reduce the duration of failure. We model a network as a collection of sites and links, each failing and recovering independently according to a Poisson process. Using this model, we demonstrate via simulation that the duration of failure incurred using a non-replicated data object is nearly as short as that incurred using a replicated object and a replication control protocol, including an unrealizable protocol which is optimal with respect to availability. We then examine analytically a simplified system in which the sites but not the links are subject to failure. We prove that if each site operates with probability p, then the optimal replication protocol, Available Copies[5, 26], reduces the duration of failure by at most a factor of $\frac{1-p}{1+p}$. Lastly, we present bounds for general systems, those in which both the sites and the communication between the sites may fail. We prove, for example, that if sites are 95% reliable and a communication failure is sufficiently short (either infallible or satisfying a function specified in the paper) then replication can improve the duration of failure by at most 2.7% of that experienced using a single copy. These results show that replication has only a small effect on the duration of failure in present-day partitionable networks comprised of realistically reliable components.

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

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 previous work we showed that, due to the database consistency constraints, replication does not increase substantially the probability that the object is available[23, 24]. In this paper we study duration of failure, which we define as the length of time, once the object becomes unavailable from a particular site, that the object remains unavailable.

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 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 we call each of the connected components resulting from the partitioning *blocks*.

It is this potential for partitioning which necessitates a replica control protocol. Given both read and write requests for access to a database object, the primary goal of a replica control protocol is to maximize the availability of the data object while ensuring the consistency of the database. Minimizing duration of failure is a useful step towards maximizing availability, not to mention retaining satisfied database users.

In order to characterize the effects of replication on the duration of failure, we first compare, via simulation, the performance of four replication control protocols to the performance of a single copy. We find that although replication always reduces the duration of failure, this reduction is relatively small. Even an off-line, oracle protocol which is optimal with respect to availability produces only a small reduction in the duration of failure.

We then develop an exact expression for the duration of failure of a single copy in a network with fallible sites and infallible communication. Using this expression, we compare the performance of a single copy to that of a replication control protocol, Available Copies[5, 26], which is optimal with respect to availability. Again we find that, given reasonably high site reliabilities, the effect on duration of failure is relatively small. Stated precisely, the optimal Available Copies protocol reduces the duration of failure by a factor of at most $\frac{1-p}{1+p}$. Thus if a site is 95% reliable, the Available Copies replication control protocol will reduce the duration of failure by at most a factor of only 2.6%

Lastly, we develop an expression for the duration of failure for a single copy in a network where both sites and links can fail. Since this expression is rather complex, we develop a number of simpler upper bounds on the duration of failure produced using a non-replicated object. Assuming, as is argued in this paper, that an optimal replication protocol with respect to availability in a partitionable network will achieve duration of failure close to the site mean time to repair, then we obtain an

interesting bound. For example, if sites are $p = 95\%$ reliable and if a communication failure from the site A to the data object at site B is sufficiently short (as specified by equation (6) of section 6.2), then replication can improve the duration of failure by at most 2.7%.

The next section describes both our network and database model and the availability and duration of failure metrics. Section 3 then describes the replica control protocols for which we demonstrate the performance in section 4. Sections 5 and 6 present bounds for systems with perfect communication and imperfect communication, respectively.

2 Model & Metrics

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 empirical results hold, but it suffices for the purposes of this paper to postulate Poisson processes for failures and recoveries[8, 33]. This assumption then allows the analytic analyses of sections 5 and 6.

We also make a possibly less realistic assumption with regard to sites and links, namely that they are fail stop[15, 35]. 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. This fail stop behavior can be approximated in software[7].

In general there will be copies of items at various sites, and there will be a stream of read and write 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. This property is called one-copy serializability[37]. In a non-replicated system, obtaining serializability is not trivial, but for the purposes of this paper, we will assume that the problem of serializability in a non-replicated system is well understood and solved, as indeed it is[6]. In unreliable, replicated systems studied in this paper, an addition protocol, called a *replica control protocol*, is needed to ensure that the value reported for a replicated object is independent of the partitioning of the network. We further define the requirements for a replica control protocol in section 3.

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 of a database management system, we assume in this paper, is the maximization of availability, given a network and the distributions just mentioned.

Although the primary object of a management system is the maximization of availability, the user is also concerned with the *duration of failure*. We define the duration of failure in a network to be the expectation over all sites of the length of time, once an access request has first been denied, that the access request will continue to be denied. The duration of failure is a function of the same system characteristics as is availability and is the object of study in this paper.

3 Protocols

Because of the partitioning alluded to above, there must be some replica control protocol, distributed over sites, 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 current because it has been changed elsewhere.

We call the block of a partition in which write accesses are allowed the *distinguished write block* and a partition in which read accesses are allowed the *distinguished read block*. A necessary rule for any correct protocol to enforce, called the *read-write consistency constraint*, is that every read distinguished block and every write distinguished block must have at least one copy in common with the most recent write distinguished block.

The protocols described below and consequently the simulations of section 4 do not differentiate between read and write requests. Therefore the read-write consistency constraint can be enforced using the *update-only consistency constraints*, which are (1) there may be at most one distinguished block at a time, and (2) successive distinguished blocks must have at least one copy in common.

3.1 Realizable Protocols

In this section we introduce a number of protocols that have been proposed to maximize availability while guaranteeing the update-only consistency constraints. Unlike the Oracle protocol presented in section 3.2, 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[4, 9, 10, 11, 13, 16, 19], we describe only those which we employ to illustrate points in this paper. Comparison with the Oracle protocol ensures that this collection of protocols is sufficient, since no protocol can perform better than the Oracle.

We call the trivial protocol governing access to a non-replicated data object the **Single Copy** protocol, *SC*. The **Primary Copy** protocol[2], 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. Since the single copy protocol and the primary copy protocol exhibit the same availability and duration of failure characteristics, we only refer to the single copy protocol. In some networks, including certain networks studied in this paper, determining the optimal location of the single or primary copy is easy. Although finding the optimal location is in general $\#P$-Complete, there exists an effective, practical method for approximating this location based upon the past performance of the system.[21]

The **Majority Consensus** protocol[36], *MC*, is an instance of the Quorum Consensus protocol[13], 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[3, 28]. For the analysis in section 4, 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 **Dynamic Voting** protocol[17, 20, 32], *DV*, is equivalent to an earlier protocol by Davčev and Burkhard[9] except that *DV* 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[17, 20] and by Long and Pâris[32].

The Dynamic Voting protocol assigns votes to each copy as does the Majority Consensus protocol, but *DV* 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[1], 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

[1] We use the Linear Ordered Dynamic Voting enhancement as described in [18]. This version of *DV* increases availability by assigning a linear ordering to the copies and using this ordering to select as distinguished one of two blocks of equal size.

copy in common. Thus the consistency constraints are fulfilled.

The **Available Copies** protocol[5], AC, unlike all the other protocols described in this section, is only applicable to networks with infallible communication. In such a system, if site A cannot communicate with site B, then site A can correctly conclude that site B has failed. Using the available copies protocol, the data object is available so long as at least one site is operational. As originally described, recovery requires waiting for every site to repair and then determining, using version numbers, the site with the latest update. We employ an improvement to this protocol by Long and Paris which modifies the recovery algorithm so that recovery occurs as soon as the site with the latest update is repaired[26]. The Available Copies protocol so enhanced has been proven optimal with respect to availability[14].

3.2 Oracle Protocols

In previous work[22], 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 such a protocol (which of course we don't know how to implement on-line) an *oracle protocol.*

In this paper, we employ two such oracles. **OracleA**, like the oracle described in [22], chooses the sequence of distinguished blocks which maximizes availability as defined in section 2. **OracleD** instead minimizes the duration of failure. Clearly, any availability obtained with the OracleA is an upper bound and any duration of failure obtained with OracleD is a lower bound for the corresponding metric for any correct protocol when the other factors are fixed.

4 Simulation

4.1 Model

In addition to the general model described in section 2, our simulator requires a number of other, more specific system parameters. 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 availability figures with a 95% confidence interval with an interval size of at most $\pm 0.5\%$ of the mean reported. Duration of failure figures reflect 95% confidence with interval size at most $\pm 3\%$ of the mean reported.

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 the networks on which simulations have hitherto been reported in the literature dealing with consistency control. A complete description of our simulator can be found in [22].

4.2 Results

From the duration of failures shown in Figure 1 and the corresponding availabilities shown in Figure 2, we make four observations and then discuss each in turn.[2]

- The duration of failure of OracleD is significantly less than that of all the other protocols.
- The Majority Consensus protocol frequently provides duration of failure shorter than that of OracleA.
- All protocols but OracleD have the same duration of failure for Topologies 256 and 4949.
- The duration of failure of OracleA does not differ substantially from that of the other realizable protocols, except in Topology 1.

The most striking observation from Figure 1 is that OracleD, which is the oracular protocol that minimizes duration of failure, provides a duration of failure roughly half that of all the other protocols. This indicates that there may be a way to significantly decrease the duration of failure of the other, realizable protocols. In fact there is such a way. Unfortunately, as Figure 2 reveals, this reduction of duration of failure is at the cost of a substantial increase in availability, which is not an attractive exchange. We consider the primary objective of distributed database replica control protocol to be the granting of as many access requests as possible while always maintaining the consistency of the database. All other performance measures, including duration of failure, are subordinate to availability. Therefore, we are not interested in decreasing the duration of failure if that requires a large decrease in availability.

The case against OracleD is strengthened once we examine the method by which OracleD reduces the duration of failures. Suppose that a distinguished block A partitions into two blocks A_1 and A_2, and then A_1 and A_2 rejoin into the single

[2] We restrict ourselves to observations based primarily on duration of failure and only secondarily on availability. We direct the interested reader to [22] for a complete discussion of the effects of replication on availability.

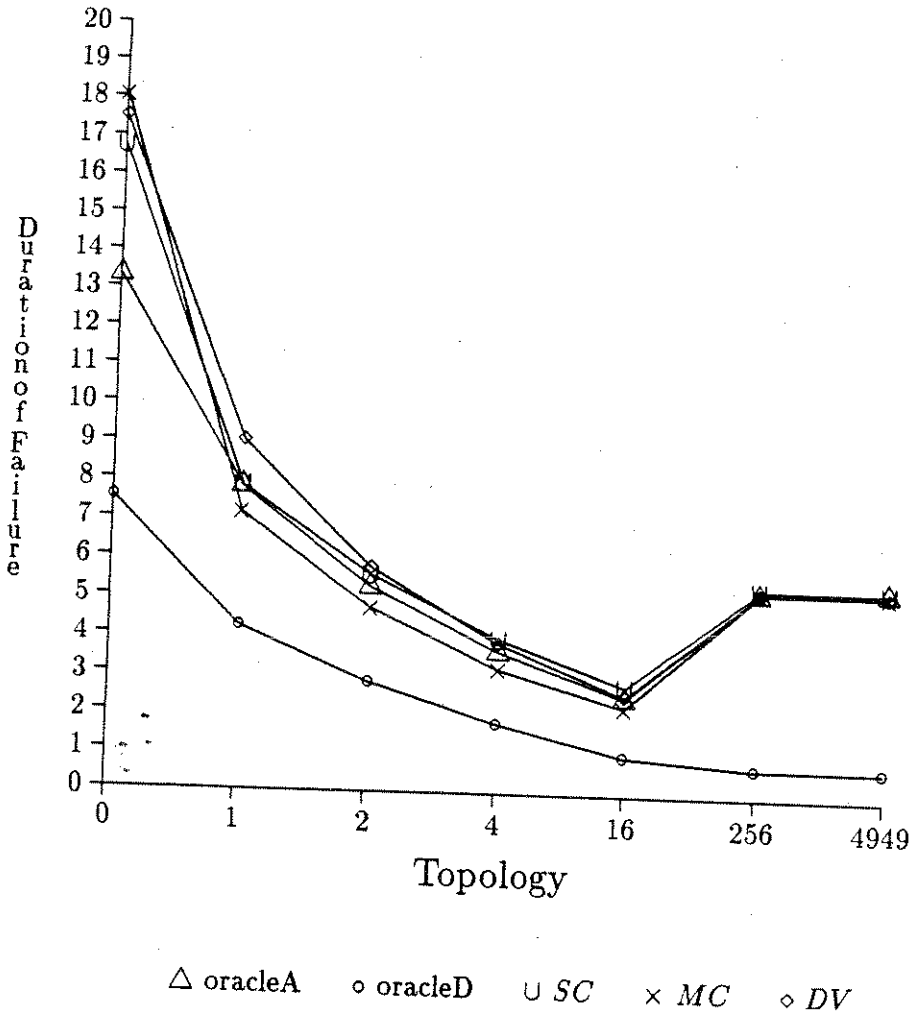

Figure 1: Duration of Failure

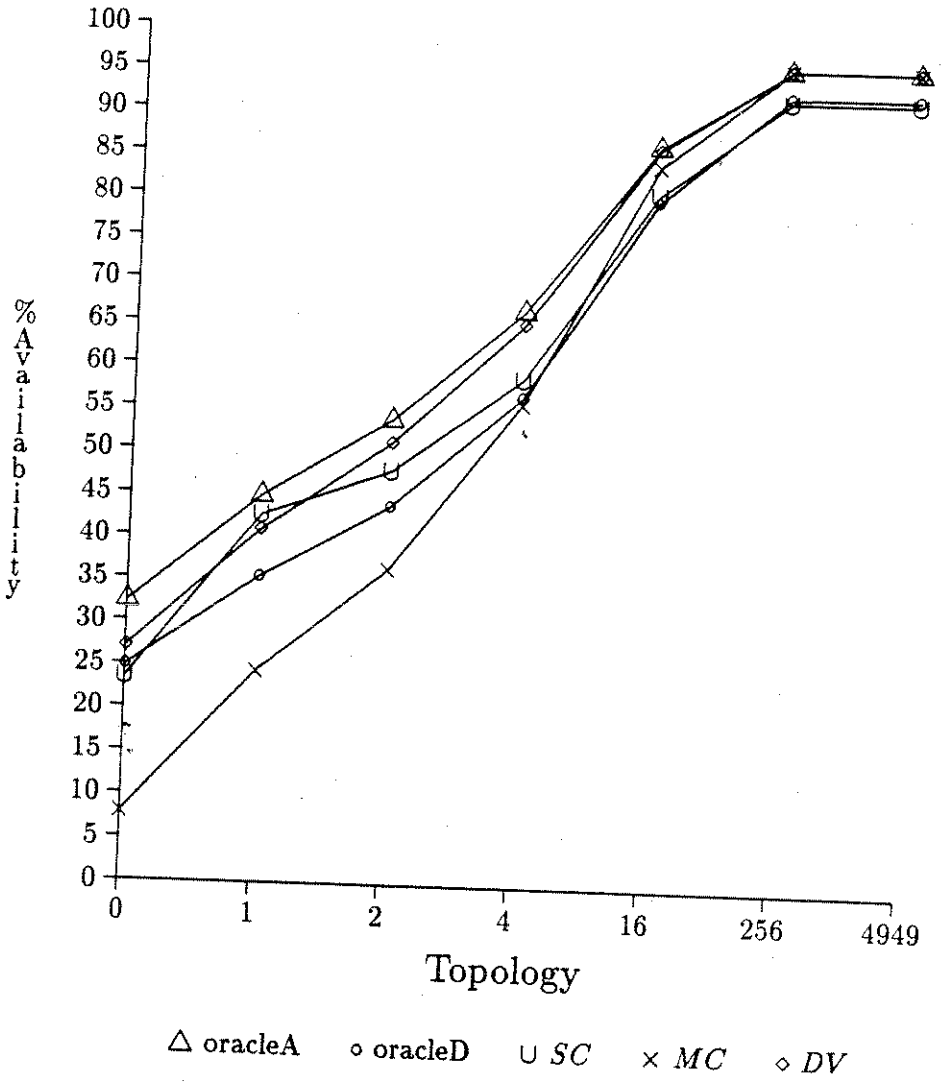

Figure 2: Availability

block A. Suppose further that the number of sites in A_1 is greater than the number of sites in A_2. Each protocol must select which of A_1 and A_2, if either, to deem distinguished. Assuming a uniform access request distribution, OracleA and DV will select A_1, since it is more probable that a larger fraction of the access requests will be submitted to the larger block than the smaller block. Likewise, MC and SC, are more likely to select A_1. OracleD, on the other hand, will choose between A_1 and A_2 based upon not only there relative size but also the duration of the partition. If that duration is longer than the average duration of failure experienced thus far, then OracleD will minimize the negative effect of this failure on the overall duration of failure by deeming A_1 distinguished, as do the other protocols. If, instead, the duration of this partition is shorter than average, then OracleD will deem the smaller block, A_2, distinguished in order to reduce the overall duration of failure by encurring a short duration of failure for the larger number of sites. If the size of A_2 is very small, for example one failed site, as compared to A_1, then availability will suffer. This explains why the availability of OracleD is poor and why a realizable protocol model after OracleD is of little value.

Our second observation, noting the frequent superiority of Majority Consensus over OracleA in terms of duration of failure, can be discounted similarly. The availability provided by MC is far too low to consider employing MC as a means of managing replicated copies.

The third observation states that all protocols but OracleD have the same duration of failure for Topologies 256 and 4949, which we call the highly connected topologies. The duration of failure to which all these protocols converge is $5\frac{1}{3}$ time units, the site MTTR. Since the topologies are highly connected, an access request succeeds virtually whenever the request site is operational, resulting in a duration of failure equal to the site MTTR.[3] The importance of this observation will be made clear in sections 5 and 6, where we argue that the optimal replica control protocol is likely to achieve duration of failure equal to the site MTTR.

Our last observation implies that replication does not substantially improve the duration of failure over that obtained with a single copy for any topology other than the ring. Although some potential for reduction of duration of failure exists in a ring topology, as evidenced by the performance of OracleA on Topology 0, none of the realizable protocols obtain this potential. Since even OracleA achieves availability less than 33%, this ring topology is of little practical interest. For the remainder of the topologies, the duration of failure of a single copy is within .33 time units or 10% of the duration of failure of OracleA.

Although these small reductions in duration of failure do not seem sufficient justification for the added communication and storage costs incurred by replication, ultimately this decision must be made by the database designer or administrator. In order to provide practical guidelines for a wide variety of topologies, protocols,

[3] As we prove formally in [24], the Majority Consensus protocol has negligible difficulty obtaining a majority of the sites in a large, highly connected network.

and read-write ratios, we present the analytic results of the next two sections.

5 Perfect Communication Networks

Simulation has provided us with empirical evidence that the contribution of replication to decreasing the duration of failure is small. In this section we derive a formula for the duration of failure for a computer network with infallible communication and a single copy. When we compare this to the duration of failure expected of the optimal replication control protocol, Available Copies, we again see that replication is of little assistance in reducing the duration of failure. Other studies in which communication has been considered infallible include [1, 5, 20, 25, 26, 28, 29, 30, 31, 34].

Let r be the mean time to repair (MTTR) for each site.

Let f be the mean time to failure (MTTF) for each site.

We begin by analyzing the duration of failure of the Available Copies protocol, since this replication control protocol is optimal with respect to availability and therefore provides the basis of comparison. If the access request is a read request then the duration of failure is clearly r, since the local copy can be read whenever the local site is operational.[4] A write access request submitted to a site Y may fail for one of two reasons. Firstly, Y may be failed, and we expect to wait r units of time for Y to repair. Secondly, all sites may have failed and, although Y has repaired, the site X with the most current version of the data object may still be failed. In this case, we also expect to wait r units of time for X to repair. Since the current copies may fail while Y is repairing and Y may fail while a current copy is repairing, the duration of failure for the Available Copies protocol is at least r. [5]

Having derived a lower bound for the duration of failure in a replicated system, we now derive an expression for the duration of failure in non-replicated systems.

Let D_1 be the expected time for both of two sites to repair given that one site is failed and the other is operational.

Let D_2 be the expected time for both of two sites to repair given that both are failed.

For two sites to repair, we must wait $r/2$ for the first to repair and then another D_1 for the second to repair. Therefore,

$$D_2 = \frac{r}{2} + D_1$$

[4] We assume that the time to update a site after repair is negligible whenever a current copy exist on an operational site. If a current copy does not exist on an operational site, then the analysis is similar to that for a write access request and also results in a duration of failure greater than or equal to r.

[5] We are satisfied with a lower bound, since the exact formulation would differ very little and would require an involved Marcovian analysis similar to [14, 26]. In fact, the difference between the bound and the exact value quickly approaches zero as the size of the network increases.

For both sites to become operational given that exactly one site is operational, we must wait $rf/(r+f)$ for the next failure or repair event. This event is a repair with probability $f/(r+f)$, and if so, both sites have repaired. If instead (with probability $r/(r+f)$) the next event is a failure, then we have two failed sites and, since the events are exponentially distributed, we must wait D_2. Therefore, if we let $p = f/(r+f)$ be the reliability of a site and use D_2 to solve

$$D_1 = \frac{rf}{r+f} + \frac{r}{r+f} D_2$$

for D_1 , we obtain

$$\begin{aligned} D_1 &= r + \frac{r^2}{2f} && (1) \\ &= r + \left(\frac{1-p}{p}\right)\frac{r}{2} && (2) \end{aligned}$$

Formula 2 assumes that the site receiving the request and the site containing the data object are not the same. Clearly, if the request is submitted to the database site, then the duration of failure is simply r. Therefore, if we let n be the number of sites in the network, and assume that the same fraction of requests are submitted to each site, then the expected duration of failure for the entire network, D_n is

$$\begin{aligned} D_n &= r + \left(\frac{n-1}{n}\right)\frac{r^2}{2f} \\ &= r + \left(\frac{n-1}{n}\right)\left(\frac{1-p}{p}\right)\frac{r}{2} \\ &\leq D_1 \end{aligned}$$

Table 1 contains some examples of the equation 2 for a variety of site reliabilities. In addition to anecdotal evidence from our own system, extensive studies have shown 95% and greater to be reasonable values for highly reliable components[4, 8, 26, 27]. Thus, for systems with realistically high site reliabilities over 95%, Table 1 shows that the reduction in duration of failure obtained using an optimal replication control protocol is less than 2.6% of that obtained using a single copy. The benefits of replication are even smaller for small networks.

6 Imperfect Communication Networks

In this section we perform essentially the same analysis for networks with fallible communication. Although the analysis is a bit more complex, the conclusion is about the same – replication does not decrease the duration of failure substantially over a single copy.

Row No.	r	f	p	(2)
1	1.0	99.0	.99	1.005
2	1.0	19.0	.95	1.026
3	1.0	9.0	.90	1.056
4	1.0	1.0	.50	1.500
5	1.0	0.5	.33	2.000

Table 1: The exact duration of failure as expressed by equation 2 for a variety of site reliabilities given perfect communication. The units are unspecified. r is both the site MTTR and the lower bound on AC duration of failure. f is the site MTTF, and $p = \frac{f}{r+f}$ is the site reliability.

6.1 Deriving Duration of Failure

Suppose that site X contains the data object and site Y receives an access request. Clearly, if $X = Y$ and the access is denied by the single copy protocol, then the duration of failure is r, the site mean time to repair. For the remainder of this section, we assume that the probability that $X = Y$ is negligible, as it would be in large networks or systems in which requests cannot be submitted to the database site. When $X = Y$ the performance of the single copy protocol can only improve with respect to duration of failure.

Let r and f be the MTTR and MTTF, respectively, for each site. Let r_c and f_c be the MTTR and MTTF, respectively, for the communication between X and Y and assume that both communication failures and repairs are exponentially distributed.[6] For the remainder of this section, we will formulate duration of failure, not as the expectation over all sites in the network, but as the expected duration of failure of a request submitted at site Y given a nonreplicated database at site X. We do this because r_c and f_c will very from site to site.

In order to compare the duration of failure of a single copy to that of a replicated copy, we must derive an upper bound for duration of failure for the single copy protocol and a lower bound for the duration of failure of a replication control protocol in partitionable network. We discuss the latter first.

As we develop a lower bound on the duration of failure, we must again deal with the following paradox which we also encountered in section 4: the duration of failure can be *reduced* to an arbitrary value by introducing *additional* failures. Since the duration of failure is an expected value, i.e. an average, adding any new failure of duration shorter than average will reduce the average. Clearly, adding failures will not achieve our primary goal of maximizing availability, nor will it

[6] It is appropriate to approximate the failure and repair time distributions of the communication system by the exponential distribution given our assumption that the failure and repair times of the constituent sites and links are exponentially distributed. See [12, pages 94–104], for example.

satisfy the database user. Instead of adding additional failures, the replication protocol must seek to reduce the duration of unavoidable failures already present in the system. Although replication can reduce or eliminate failures due to inaccessible communication paths or failed sites containing copies, our model of access request submission implies that the failures incurred when the submission site is failed are unavoidable. These unavoidable failures are of duration r, the site MTTR.

A lower bound for duration of failure of r, the site MTTR, is especially reasonable for *read-intensive* databases and for highly reliable networks. A read-intensive database is one in which the vast majority of access requests are read requests. Optimal availability in such a database is usually obtained using the **Read-one/Write-all** protocol[23], which requires that a read be performed only on the local copy, but that a write be performed on every copy in the network. Clearly, a failure is most likely to be a read failure in a read-intensive database, and a read failure is of duration at least r. The same is true for networks with highly reliable communication, since the performance of such systems approaches that described in section 5. There the lower bound is provably equal to r.

Therefore, we conclude that an optimal replication control protocol in a partitionable network is likely to have duration of failure at least r, the site MTTR. We now formulate an expression for the duration of failure of a single copy in such a network. We will conclude this section by showing that the lower bound for the optimal replication control protocol is not substantially less than the upper bound for the single copy protocol.

Let D_{XYC} be the duration of failure for the single copy protocol when the network is in the state represented by XYZ, where $X = u$ if the database site X is operational (up) and $X = d$ (down) otherwise, $Y = u$ if the submit site Y is operational and $Y = d$ otherwise, and $C = u$ if the communication from site X to site Y is operational and $C = d$ otherwise. For example, D_{uud} is the duration of failure given that both X and Y are operational and the communication is down. Since the reliability characteristics are assumed to be the same for X and Y, $D_{duu} = D_{udu}$ and $D_{dud} = D_{udd}$. We use D to denote the duration of failure experienced at site Y as a result of any one of these failure conditions, still assuming that $X \neq Y$.

Using this notation, the duration of failure, D, under the single copy protocol can be expressed as the probability that site X is down times D_{duu}, plus the probability that site Y is down times D_{udu}, plus the probability that the communication is down times D_{uud}. Since the reliability characteristics are assumed to be the same for X and Y, this can be expressed as

$$D = \frac{2ff_c}{f^2 + 2ff_c} D_{duu} + \frac{f^2}{f^2 + 2ff_c} D_{uud} \tag{3}$$

Solving the following system of equations for D_{duu} and D_{uud}

$$D_{duu} = \frac{rff_c}{rf + rf_c + ff_c} + \frac{rf_c}{rf + rf_c + ff_c} D_{ddu} + \frac{rf}{rf + rf_c + ff_c} D_{dud}$$

$$
\begin{aligned}
D_{ddu} &= \frac{r^2 f_c}{r^2 + 2rf_c} + \frac{2rf_c}{r^2 + 2rf_c} D_{duu} + \frac{r^2}{r^2 + 2rf_c} D_{ddd} \\
D_{ddd} &= \frac{r^2 r_c}{r^2 + 2rr_c} + \frac{2rr_c}{r^2 + 2rr_c} D_{dud} + \frac{r^2}{r^2 + 2rr_c} D_{ddu} \\
D_{uud} &= \frac{f^2 r_c}{f^2 + 2fr_c} + \frac{2fr_c}{f^2 + 2fr_c} D_{dud} \\
D_{dud} &= \frac{rfr_c}{rf + rr_c + fr_c} + \frac{fr_c}{rf + rr_c + r_c f} D_{uud} + \\
&\quad \frac{rf}{rf + rr_c + r_c f} D_{duu} + \frac{rr_c}{rf + rr_c + r_c f} D_{ddd}
\end{aligned}
$$

we obtain

$$D = \frac{2rff_c + r^2 f_c + f^2 r_c + 2rfr_c + r^2 r_c}{f^2 + 2ff_c} \tag{4}$$

If we let $p = \frac{f}{r+f}$ be the reliability of a site and $p_c = \frac{f_c}{r_c+f_c}$ be the reliability of communication, then the duration of failure can be rewritten as

$$D = \frac{1 - p^2 p_c}{p(pr - pp_c r + 2p_c r_c - 2pp_c r_c)} rr_c \tag{5}$$

Table 2 shows the value of D for a number of system configurations. Note that when the site reliabilities are over 95% and communication reliabilities are over 90%, as one would expect using today's components and technologies[4, 8, 26, 27], D is within 6.8% of r. Since we expect the optimal replication control protocol to have duration of failure at least r, in these examples, replication decreases the duration of failure in these instances by at most 6.4% of the duration of failure of a single copy.

6.2 Bounding Duration of Failure

The equation D is a bit cumbersome, so we include two upper bounds on D. From the derivative of D with respect to p_c, the reliability of communication, we see that D is a monotonically increasing function of p_c iff $pr + p^2 r > r_c$, that D is a monotonically decreasing function of p_c iff $pr + p^2 r < r_c$, and that D is independent of p_c when $pr + p^2 r = r_c$. Since $0 \leq p_c \leq 1$,

$$D \leq \begin{cases} \frac{1}{p^2} r_c & \text{if } r_c > \frac{1}{2}(pr + p^2 r) \\ \frac{1+p}{2p} r & \text{otherwise} \end{cases} \tag{6}$$

The column headed (6) of Table 2 contains the values for equation 6 for a number of selected system configurations.

Row No.	Site			Communication			Formulas		
	r	f	p	r_c	f_c	p_c	(5)	(6)	(7)
1	1.0	99.0	.99	0.5	49.5	.99	0.758	1.005	1.005
2	1.0	99.0	.99	1.0	99.0	.99	1.010	1.020	1.020
3	1.0	99.0	.99	1.5	148.5	.99	1.136	1.530	1.530
4	1.0	19.0	.95	0.5	4.5	.90	0.706	1.026	1.026
5	1.0	19.0	.95	1.0	9.0	.90	1.068	1.108	1.108
6	1.0	19.0	.95	1.5	13.5	.90	1.289	1.662	1.662
7	1.0	19.0	.95	0.95	8.6	.90	1.040	1.053	1.108

Table 2: Each formula for networks with imperfect communication instantiated with different site and communication reliability parameters. The units of are unspecified. r is both the site MTTR and the lower bound on an optimal replication control protocol duration of failure. f is the site MTTF, and $p = \frac{f}{r+f}$ is the site reliability. r_c, f_c, and p_c are defined similarly for the communication reliability parameters. 5 is an exact formula and 6 and 7 are upper bounds for the duration of failure of a single copy.

Since we can only express the duration of failure of an optimal replication protocol in terms of r, it would be helpful if equation 6 where in terms of r rather than r_c. Thus D can be bound, although more loosely than 6, by

$$D \leq \begin{cases} \frac{1}{p^2} r_c & \text{if } r_c > r \\ \frac{1}{p^2} r & \text{if } r \geq r_c > \frac{1}{2}(pr + p^2 r) \\ \frac{1+p}{2p} r & \text{otherwise} \end{cases} \qquad (7)$$

The column headed (7) of Table 2 contains the values for equation 7 for a number of selected system configurations. In particular, notice Row 7 where the value for bound 7 differ from that for bound 6. Although the value for the latter bound is greater, the latter bound may be of value since it is not dependent upon exact communication reliability characteristics.

Although the duration of failure can be bound in a number of other ways, the most useful form is a function of the known system parameters, which vary from system to system. For many systems, it may be helpful to replace $p^2 p_c$ with the availability of a single data object from a particular site.

7 Conclusions

We have shown, both analytically and via simulation, the limitations of replication with respect to duration of failure. Our simulation results indicate that replication does not dramatically reduce the duration of failure when compared to protocols realizable in real networks and to an unrealizable protocol that is optimal with respect to availability. We also developed upper bounds on the performance benefits of replication in networks with infallible communication and exponentially distributed MTTR and MTTF for sites. We proved that if each site operates with probability $p = \frac{MTTF}{MTTF+MTTR}$, then the optimal replication protocol, Available Copies, reduces the duration of failure by at most a factor of $\frac{1-p}{1+p}$. We also presented bounds for systems in which both the sites and the communication between the sites may fail. These results show that replication has only a small effect on the duration of failure in present-day partitionable networks comprised of realistically reliable components.

Clearly, data replication may prove valuable according to other measures, for example, communication cost and response time for read-intensive databases. On the other hand, these result elucidate the limitations of data replication with respect to duration of failure. Other studies have shown similar limitations with respect to availability[23, 24]. It is our hope that this work will assist database developers and researchers in applying resources to those ends for which they are best suited.

References

[1] 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.

[2] 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.

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

[4] 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.

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

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

[7] Walter A. Burkhard, B.E. Martin, and Jehan-François Pâris. The gemini fault-tolerant file system: the management of replicated files. In *Proceedings of the Third International Conference on Data Engineering*, pages 441–448. IEEE, 1987.

[8] 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.

[9] 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.

[10] 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.

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

[12] I. B. Gertsbakh. *Statistical Reliability Theory.* Marcel Dekker, Inc., New York, 1989.

[13] David 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.

[14] Alexander Glockner. Optimal consistency protocols for replicated files. In *8th Annual International Phoenix Conference on Computers and Communications*, pages 434–438. IEEE, March 1989.

[15] Jim Gray. Why do computers stop? Tandem Computers, P#87614, Cupertino, CA, 1985.

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

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

[18] 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, September 1987.

[19] 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.

[20] 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.

[21] Donald B. Johnson and Larry Raab. Complexity of network reliability and optimal database placement problems. Technical Report PCS-TR91-167, Dartmouth College, June 1991.

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

[23] 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.

[24] 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.

[25] 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.

[26] Darrell D. E. Long and Jehan-François Pâris. 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.

[27] Darrell D. E. Long, Jehan-François Pâris, and C. J. Park. A study of the reliability of internet sites. Technical Report UCSC-CRL-90-46, University of California, Santa Cruz, September 1990.

[28] 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.

[29] Jerre D. Noe and Agnes Andreassian. Effectiveness of replication in distributed computer networks. In *Proceedings of the Seventh International Conference on Distributed Computing Systems*, pages 508–513. IEEE, September 1987.

[30] Mirjana Obradovic and Piotr Berman. Voting as the optimal static pessimistic scheme for managing replicated data. In *Ninth IEEE Symposium on Reliable Distributed Systems*, pages 126–135, 1990.

[31] Jehan-François Pâris. 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.

[32] 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.

[33] Jehan-François Pâris, Darrell D. E. Long, and Alexander Glockner. A realistic evaluation of consistency algorithms for replicated files. In *The 21st Annual Simulation Symposium*, pages 121–130. IEEE, March 1988.

[34] Calton Pu, Jerry D. Noe, and Andrew Proudfoot. Regeneration of replicated objects: A technique and its eden implementation. In *Proceedings of the Eighth ACM Symposium on Operating Systems Principles*, pages 169–177. ACM, December 1981.

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

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

[37] I. L. Traiger, J. N. Gray, C. A. Galtier, and B. G. Lindsay. Transactions and consistency in distributed database systems. *ACM Transactions on Database Systems*, 7(3):323–342, June 1982.