Estimating the Reliability of Regeneration-Based Replica Control Protocols

Darrell D.E. Long
Computer and Information Sciences
University of California
Santa Cruz

John L. Carroll
Kris Stewart
Department of Mathematical Sciences
San Diego State University

IEEE Transactions on Computers, vol. 38, no. 12, December 1989, pp. 1691–1702.

Abstract

The accessibility of vital information can be enhanced by replicating the data on several sites, and employing a consistency control protocol to manage the replicas. The reliability of a replicated data object depends on maintaining a viable set of current replicas. When storage is limited, it may not be feasible to simply replicate a data object at enough sites to achieve the desired level of reliability. *Regeneration* approximates the reliability provided by additional replicas for a modest increase in storage costs, and is applicable whenever a new replica of a data object can be created faster than a system failure can be repaired. Regeneration enhances reliability by creating new replicas on other sites in response to site failures.

Several strategies for replica maintenance are considered, and the benefits of each are analyzed using simulation and both algebraic and numeric solutions to systems of differential equations. Formulas describing the reliability of a replicated data object are presented, and closed-form solutions are given for the tractable cases. Numerical solutions, validated by simulation results, are used to analyze the trade-offs between reliability and storage costs. The space savings attributable to regeneration are also quantified.

1 Introduction

Distributed systems can afford enhanced fault tolerance through redundancy, and the evaluation of the increased performance is of great practical interest. The most common measures of fault tolerance include *reliability,* which is the probability that a replicated data object will remain continuously available over a given time period, and *availability,* which is the steady-state probability that the data object is available at any given moment. Availability has received much more attention, in part because its analysis is more tractable than that of reliability.

There are several reasons for favoring reliability as the primary performance measure. For comparable numbers of sites, the availabilities afforded by the better protocols are very similar, but the reliabilities vary greatly. In many applications, the reliability of a system is a more important measure of its performance than its availability. These applications include process control, data gathering, and other tasks requiring interaction with real-time processes, where the data will be lost if not captured when it is available. The computers used for stock trading are a prime example: If these machines were to fail, the resulting chaos would halt trading.

The accessibility of vital information can be enhanced by replicating the data on several sites, and employing a consistency control protocol to manage the replicas. A replicated data object has the same semantics as an unreplicated instance of the same data object. The reliability of a replicated data object depends on maintaining a viable set of current replicas. Costs or space limitations may make it impossible to replicate a data object at enough sites to guarantee an acceptable level of fault tolerance. If new replicas of a data object can be created faster than a system failure can be repaired, then better reliability can be achieved by creating new replicas on other sites in response to site failures. This technique, known as *regeneration,* approximates the protection provided by additional replicas with only a modest increase in storage costs.

The notion of regenerating replicas to replace those lost due to site failures was first proposed by Pu [23, 24]. His protocol, called the *Regeneration Algorithm,* provides mutual and serial consistency of replicated data objects in a partition-free distributed system. The Regeneration Algorithm is simple and efficient, but some weaknesses can be identified. It allows reads to continue as long as one current replica of the object remains accessible, while writes are disabled whenever there are fewer than the initial number of replicas are accessible and there are insufficient spares to replace the missing replicas. The protocol specification leaves the procedure for recovering from a total system failure unspecified, requiring manual intervention in the event of a total system failure. It is also unable to guarantee mutual consistency in the presence of network partitions. This approach is suitable for single-segment carrier-sense networks, but fails on the increasingly more common multi-segment networks.

Noe and Andreassian suggested the adoption of an approach similar to the Available Copy [1, 2] protocol to alleviate the problem of poor reliability for writes since it allows writes to occur as long as a single replica of the data object remains available [20]. This suggests that using regeneration to maintain a viable set of current replicas is a technique that can be adapted to many existing replica control protocols. By combining regeneration with protocols that have desirable characteristics, the weaknesses of the Regeneration Algorithm can be successfully addressed.

The performance of several replica control protocols which use regeneration is explored in this article. It has been shown [13, 17, 15] that regeneration is a generally applicable technique that can be profitably combined with many replica control protocols. Regeneration can be used with the Available Copy protocols [1, 2, 4] to improve fault tolerance in non-partitionable computer networks. When the communications network is susceptible to partitionings, consensus protocols are required. By combining regeneration with static Majority Consensus Voting, a simple protocol for maintaining mutual consistency among replicas of a data object is obtained. Regeneration can also be combined with the Dynamic Voting protocols [6, 12, 16] to provide an increased level of fault tolerance over that of static consensus protocols.

The techniques used here to estimate performance are similar to those used to analyze protocols based solely on replica repair [21, 5, 4, 16, 18]. The added complexity due to replica replacement complicates the analysis of regeneration-based protocols, requiring the application of more powerful techniques.

2 Regeneration

Pu [23] first proposed the Regeneration Algorithm as a space-efficient technique for increasing the availability of replicated data objects in the *Eden* system [24]. New replicas are created (regenerated) when the algorithm detects that site failures have rendered one or more of the replicas inaccessible. The protocol initiates a regeneration only when a write request occurs, thereby enhancing its efficiency. There are several improvements to the Regeneration Algorithm that can be made by adapting the technique to existing replica control protocols. The hybrid protocols presented in the following sections successfully address the limitations of the original Regeneration Algorithm. When the communications network is not susceptible to partitioning, an Available Copy protocol [1, 2] provides the best fault-tolerance characteristics. The Optimistic Available Copy protocol [5, 4] is an ideal candidate for regeneration due to its efficient implementation. When communications failures can occur, a consensus-based protocol based on Dynamic Voting [6] provides a high degree of fault tolerance while protecting against replica divergence. Regeneration can be easily applied to most Dynamic Voting derivatives, including Dynamic-linear Voting [12] and Optimistic Dynamic Voting [16]. This technique can also be applied to static Majority Consensus Voting [7, 9].

Another concern often raised about regeneration is its cost in storage space. While in the worst case the amount of space required is the number of original replicas plus the number of spares, preliminary studies have shown that the increase in storage requirements is on the average less than 10 percent [14] for typical failure and repair rates.

2.1 Applying Regeneration

To combine regeneration with an existing replica control protocol, it is necessary for the protocol to differentiate between sites holding replicas and sites to be used when replicas are regenerated. The protocol must be able to identify these sites so that it can consider only those replicas that belong to the correct generation of the replicated data object. This is especially important in the case of consensus-based protocols, as it is essential to disenfranchise replicas that belong to previous generations to avoid possible inconsistencies among replicas of the data object.

A replica control protocol based on regeneration begins with a set of replicas placed on sites around the computer network. As these replicas fail, other sites (called spares) are located and new replicas are created on them by the regeneration protocol. Once a failed system component has been repaired, the storage used by the extra replicas can be relinquished.

For all of the protocols, it is essential that the regeneration mechanism be atomic to ensure consistent creation of replicas. For quorum-based protocols, generations with fewer than a quorum could result if this condition is not met. In this event, the replicated data object would become permanently inaccessible since no quorum could ever be formed. It is assumed that an appropriate commit protocol [26] is used to ensure atomicity.

2.1.1 Regenerative Available Copy

Applying regeneration techniques to an Available Copy protocol was first suggested by Noe and Andreassian [20]. An Available Copy protocol increases the reliability of the protocol by continuing to allow writes to occur as long as one replica of the data object remains available and by allowing reads from any available copy. Combining the Available Copy protocol with regeneration affords better performance than the Regeneration Algorithm, which disallows writes when less than a full complement of sites are available.

The Optimistic Available Copy protocol [5, 4] provides all the facilities that are required to integrate regeneration into an available copy protocol, and is therefore a good basis for regeneration. In particular, the was-available sets, which are used by Optimistic Available Copy to speed recovery from total failure by tracking the last site to fail, provide all the information that is needed to identify the set of replicas comprising the current generation. The was-available set for an available site represents those sites which received the most recent change to the state of the replicated data object. This includes the set of all sites that received the most recent write and all of those sites which have repaired from that site.

For the Optimistic Available Copy protocol to allow a regeneration to occur, at least one replica must be in an available state and there must be at least one spare. The state of the replicated data object, including the data and the version number, is first copied to the spare sites, and the names of those spare sites are then included in the was-available sets of all available sites. The version number associated with the regenerated replicas is the current version number of the replicated data object.

Except for its treatment of spares, the recovery procedure remains the same as in the Optimistic Available Copy protocol. It operates by using the information contained in the was-available sets stored at each site to determine the set of sites that failed last. This set of sites can be found by computing the closure of the was-available set of the recovering site. When a site recovers from a failure and finds a site in its was-available set that has been transformed into a spare, it treats that spare as a failed site. The recovering site will then be unable to compute the closure of its was-available set, ensuring that it must wait for the recovery of the last site to fail.

When a recovering site establishes communication with an available site, it may find that a full complement of replicas are already available. In this case, one of the extant replicas can be destroyed and the storage reclaimed, thus reducing the amount of storage consumed. As with all regeneration-based protocols, the question of which replicas should be retained remains open. It would seem that the best choice would be to keep replicas on sites with the best reliability characteristics, but factors such as communication costs may make other choices more appropriate.

To show that Optimistic Available Copy with regeneration is correct, it is necessary and sufficient to prove that the replicated data object will not be made available following a total failure of the system until the last site to fail can be determined. This is sufficient because of the assumption that the communication network will not partition, preventing inconsistencies due to competing accesses. The following theorem demonstrates this result.

Theorem 2.1. *When using the Optimistic Available Copy protocol with regeneration, a site recovering from a total failure will not enter an available state until the last site to fail has been found.*

Proof: There are two cases to consider. The first case occurs when a recovering site is able to compute the closure of its was-available set. The result then follows directly from the proof of correctness for Optimistic Available Copy [4], since being able to compute the closure of its was-available set implies that there are no reclaimed spares in the chain of successors of the site. Thus, the last site to fail can be determined by examining the version number of each of the replicas.

The second case occurs when a site is unable to compute the closure of its was-available set; this must be because there is a reclaimed spare which is being treated as a failed site. In this case, the site could not have been the last site to fail since if it were, then there could be no reclaimed spare as its successor. Consequently, the site must wait for the last site to fail; this last site will be able to recover since it has no reclaimed spare as a successor. □

2.1.2 Regenerative Majority Consensus

It is a simple matter to combine regeneration with static Majority Consensus Voting. It requires the notion of generations in order to determine the current set of replicas. A generation is defined as the set of replicas that have participated in a particular regeneration. Each replica in the set will be tagged with a *generation number*. By using generations, the current set of replicas can easily be determined, and only that group is allowed to participate in quorum collection. The generation numbers used here are similar to those used by Pâris in his article on *voting with tokens* [22].

Protocol 2.1 is a simple extension of static Majority Consensus Voting [7, 9]. A majority is considered to be a majority of the votes assigned to the original set of replicas. As spares replace failed sites they can be assigned the votes of the failed sites. In this case, the net number of votes in any generation is never more than the original number of votes. Of course, it is possible to reassign votes using an appropriate protocol, but that issue is orthogonal to this discussion.

When the protocol determines that there are fewer than the desired number of replicas of the data object, a regeneration will be initiated. A regeneration cannot occur unless a quorum of the replicas can be collected. These replicas must be members of the current generation, as indicated by the current generation number. If a quorum exists and there are spare sites available, some are chosen to hold the new replica of the data object. The state of the replicated data object, including the data and the version and generation numbers, is copied to the spare sites. The spares are transformed into full replicas and all participating sites will increment their generation numbers in order to disenfranchise any sites that did not participate. This preserves mutual consistency by excluding the replicas that did not participate in the regeneration from taking part in any future quorum.

When a vote is called, excess replicas of the data object will have obsolete generation numbers. These replicas can be transformed into spares since they are not members of the current generation and so cannot participate in any quorum.

Protocol 2.1. *Regeneration of replicas under Regenerative Majority Consensus.*

1. *Determine all sites that can communicate in this partition of the network, call it U.*
2. *Let $R = \{i \in U : g_i = \max_{j \in U}\{g_j\}\}$. R is the most current generation of replicas that can be found. If the cardinality of R is greater than one half of the original complement of replicas, then a quorum is present and the regeneration can continue.*
3. *Determine the set of spares that are present in this partition of the network, call it S.*
4. *Choose a subset T of spares from S, up to the original complement of replicas, with the best fault-tolerance characteristics.*
5. *Copy the state of the replicated data object to each of these spares, including version numbers and generation numbers. Increment the generation numbers of each replica in $R \cup T$.*

To demonstrate the correctness of Regenerative Majority Consensus, it is necessary and sufficient to show that at any time there will be at most one generation of replicas that can constitute a quorum. The following theorem is similar to a result demonstrated by Pâris [22].

Lemma 2.1. *Under static Majority Consensus Voting with regeneration, the protocol for creating a new generation of replicas will disenfranchise a minority of the current generation.*

Proof: When a regeneration occurs, a majority of the current generation must be present. This implies that a minority of the replicas will not be participating in the regeneration. When the generation numbers are incremented, these replicas will be left with out-of-date generation numbers. □

Theorem 2.2. *Under static Majority Consensus Voting with regeneration, the only quorum present will be made up of sites with the current generation number.*

Proof: The result follows directly from Lemma 2.1. Since a minority of sites have been disenfranchised, they cannot form a quorum. A collection of minorities will be unable to merge and form a quorum since only a minority of replicas hold generation numbers other than the current generation number. □

There is no need for a complex site recovery protocol, since a site that recovers from a failure can determine if it is a member of an earlier generation by examining its generation number when the next quorum collection occurs. If it is found to be a member of an earlier generation it can be transformed into a spare.

2.1.3 Regenerative Dynamic Voting

The Optimistic Dynamic Voting protocol [16] is an implementation of Dynamic Voting [6], similar to Dynamic-linear Voting [12] but more amenable to regeneration. Due to its use of partition sets, Optimistic Dynamic Voting accepts regeneration naturally. A partition set represents the set of sites which participated in the last successful operation that included the site where it is stored. They are used to determine the required quorum for the next access operation. The partition sets are maintained when either a read or write operation occurs, and are brought up-to-date when a site recovers from a system failure.

The partition sets contain all the information that is needed to identify replicas and provide a mechanism for excluding sites that are members of out-of-date quorums. Accomplishing a regeneration requires a majority of the replicas in the quorum set to be present. The definition of a quorum remains unchanged. Spare sites are selected and are transformed into replicas and the names of these sites are entered into the partition sets of sites in the quorum. The operation number of the replicated object is then incremented. Since a quorum must be present, incrementing the operation number has the effect of disenfranchising those replicas that did not participate in the regeneration.

There is only a slight change in the site recovery protocol. When a site recovers from a failure and finds that the original number of replicas are already present, then the storage that it consumes is superfluous and this site can be transformed into a spare.

Protocol 2.2. *Regeneration of replicas under Regenerative Optimistic Dynamic Voting.*

1. *Find the set of all sites communicating with the requesting site, call it R.*
2. *Request from each site $i \in R$ its partition set P_i, its operation number o_i, and its version number v_i.*
3. *Let $Q = \{i \in R : o_i = \max_{j \in R}\{o_j\}\}$. Q is the most current generation of replicas that are available.*
4. *Let P_m be the partition set of any site in Q.*
5. *If the cardinality of Q is greater than one half the cardinality of P_m, or is exactly one half and contains the maximum element of P_m then the current partition is the majority partition and the regeneration can continue.*
6. *Determine the set of spares, S, that are present in this partition of the network.*
7. *Choose a subset T of spares from S, up to the original complement of replicas, with the best fault-tolerance characteristics.*
8. *Send $Q \cup T$ as the new partition set to each site along with the new operation number $o_m + 1$.*

To demonstrate the correctness of Regenerative Optimistic Dynamic Voting, it is necessary and sufficient to show that at any time there will be at most one generation of replicas that can constitute a quorum. This occurs because the act modifying the quorum provides the mechanism for disenfranchising sites.

Theorem 2.3. *Under Optimistic Dynamic Voting with regeneration, there is at any time at most one majority partition.*

Proof: The result is similar to that showing the correctness of the original Optimistic Dynamic Voting protocol [13]. What remains to be shown is that the regeneration protocol will not introduce any extraneous majority partitions.

Observe that a majority of the most recent quorum set must be present in order for a regeneration to occur. This implies that only a minority of the more recent quorum set are not participating in the regeneration. Incrementing the operation numbers suffices to disenfranchise this minority of sites, thus preventing it from forming a majority partition. Although the new quorum set is being enlarged, this does not affect the correctness of the protocol; it remains the only set of replicas for which a majority of the sites whose names are in the partition sets have current operation numbers. □

3 Reliability Analysis

While the availability of a protocol measures the performance of that protocol over a long period of time, its reliability estimates the probability a replicated data object managed by that protocol will remain continuously available over a given period of time. In general, the reliability $\mathscr{R}_P(n, m, t)$ of an n-site system with m spares and managed by protocol P is defined as the probability that the system will operate correctly over a time interval of duration t given that an initial complement of n replicas and m spares were operating correctly at time $t = 0$.

The requirements of the application manipulating the replicated data object affects the relative importance of the measures. While reliability is perhaps the best indicator of the dynamic behavior of the system, availability is often of primary concern if the objective is to simply minimize the inaccessibility of the data being replicated. By contrast, enhanced reliability is usually the main objective for applications that incur disproportionately high costs for any failure of the data object.

Since availability is a measure of the steady-state properties of a system, the availability of replicated data objects has been extensively studied. Reliability is a measure of the behavior of a system under transition, and its analysis is much less tractable.

Because Optimistic Dynamic Voting and Dynamic-linear Voting have the same performance, and since Optimistic Available Copy is an efficient implementation of Available Copy, Dynamic-linear Voting (DLV) and Available Copy (AC) will be used for the remainder of this article.

3.1 System Model

The replicas of the data object are assumed to reside on distinct sites of a computer network, and these sites have independent failure modes, but have identical failure, repair, and regeneration characteristics. The communication network connecting the sites is assumed to be reliable. The time to notice a site failure and complete a repair is assumed to be exponentially distributed with mean $1/\mu$. This includes the time to ascertain if this site is still intended to hold a replica of the data object and (if necessary) copy the data. Site failures are assumed to be exponentially distributed with mean rate λ. Site regeneration is similarly modeled by an exponential distribution with mean κ, which reflects the time to determine that a site has failed, verify that both the replicated data object and a suitable spare is available, and install a replica on that spare site.

In a network with a large cluster of work stations, the number of spares is often much greater than the desired number of replicas, and hence the number of spares can be viewed as being effectively unlimited. The failure of the replicated data object will rarely be due to the unavailability of a suitable spare, but will usually result from the inability of an existing spare to successfully replicate the data before the last site fails. The unlimited spare scenario is much more tractable than the general case and will be analyzed first.

The differential equations describing the behavior of systems managed by the replica control protocols can be derived from the state-transition flow rate diagrams. The states in the Markov chain are labeled to reflect the number of sites that can successfully respond to a request for the replicated data object. An n-site system with an unlimited number of spares is in state $\langle 0 \rangle$ if the replicated data object has been inaccessible at some point in the past, while for $1 \le i \le n$, the system is in state $\langle i \rangle$ if the object has been continuously accessible and if i replicas of the data object are currently accessible. Thus, no transitions are permitted from state $\langle 0 \rangle$, since only the behavior of the system prior to the first total failure is of interest. Flow rates to adjacent states are governed by the number of sites operational and the number of sites under repair. The diagram for an n-site system with an unlimited number of spares employing an Available Copy protocol is given in Figure 1.

A system maintaining n active sites with an additional m spare sites is in state $\langle j, k \rangle$ if j replicas are immediately accessible and k sites are currently available as spares. The state $\langle 0 \rangle$ will again denote the inaccessible state. The flow rate diagrams for three sites with two spares managed by Available Copy, Dynamic-linear Voting, and Majority Consensus Voting are shown in Figures 2, 3, and 4, respectively.

Definition 3.1. *The* reliability $\mathscr{R}_P(n, m, t)$ *of an n-replica system with m spares and managed by protocol P is defined as the probability that the system will operate correctly over a time interval of duration t given that an initial complement of n replicas and m spares were operating correctly at time* $t = 0$.

The equations resulting from the flow rate diagrams in Figures 2 through 4 will be used to obtain $\mathscr{R}_{AC}(3, 2, t)$, $\mathscr{R}_{DLV}(3, 2, t)$, and $\mathscr{R}_{MCV}(3, 2, t)$.

3.2 Analytic Results

For small numbers of sites, closed-form solutions for the reliability of some of the protocols can be obtained from the differential-difference equations. Less tractable systems can be both simulated and solved numerically. Simulation is crucial to characterizing site regeneration as a Poisson process: a site failure is generally discovered only after a non-trivial period of time. Since κ reflects the time necessary to both detect a site failure and restore the data base, exponential distributions are at best an approximation.

As is apparent in Figure 5, the reliability of these systems is relatively insensitive to the shape of this distribution. The data points shown on the graph were obtained by simulating the repairs and failures of a system of 3 sites and 2 spares until all sites failed, and noting the time at which the protocol would first deny access to a replicated data object. The process was repeated 1000 times, and the results were sorted to obtain an approximation of the reliability function. In this graph and those that follow, the site repair rate μ is used as the basic time unit, yielding a relative time scale where $\mu = 1$. The resulting deciles reflect several regeneration distributions with identical first moments but disparate higher moments.

The effects of higher moments of the regeneration distributions on system reliability is not unexpected; the reliability of these protocols has already been shown to be relatively insensitive to the shapes of the failure and repair distributions [3].

3.2.1 Unlimited Spares

It is only possible to derive closed-form solutions in the most elementary cases, such as when a protocol such as Regenerative Available Copy is assumed to have an unlimited supply of spares available. The set of differential-difference equations arising from n sites managed by Regenerative Available Copy with an infinite number of spares, as shown in Figure 1, is given by

$$\begin{aligned}
\frac{dp_n}{dt} &= \kappa p_{n-1}(t) - n\lambda p_n(t),\\
\frac{dp_j}{dt} &= (j+1)\lambda p_{j+1}(t) + (n+1-j)\kappa p_{j-1}(t) - (j\lambda + (n-j)\kappa)p_j(t), 1 < j < n\\
\frac{dp_1}{dt} &= 2\lambda p_2(t) - (\lambda + (n-1)\kappa)p_1(t)\\
\frac{dp_0}{dt} &= \lambda p_1(t)
\end{aligned}$$

with initial conditions

$$p_i(t) = \begin{cases} 0 & 0 \le i < n \\ 1 & i = n \end{cases}$$

In this system, $p_0(t)$ represents the probability that at time t the system has failed. The reliability of the system is $\mathscr{R}_{AC}(n,t) = 1 - p_0(t)$. When $n = 2$, the time-dependent solution to this Markov process yields

$$\mathscr{R}_{AC}(2,t) = \left(\frac{(3\lambda+\kappa)\sinh(\frac{t\sqrt{\lambda^2+6\kappa\lambda+\kappa^2}}{2})}{\sqrt{\lambda^2+6\kappa\lambda+\kappa^2}} + \cosh(\frac{t\sqrt{\lambda^2+6\kappa\lambda+\kappa^2}}{2})\right)e^{-\frac{(3\lambda+\kappa)t}{2}}$$

when failed sites are simply replaced from an unlimited supply of spares. In the presence of a total failure, no copies of the data object are available to regenerate a spare, and the replicated data object will be inaccessible until the critical sites have recovered. This does not affect the analysis of the reliability of the system, since it is only the behavior of the system prior to a total failure that is of interest.

The equations for a system in which the repair of a failed site proceeds in parallel with the regeneration of a spare site are quite similar, with each κ replaced by $\kappa + \mu$, and for $n = 2$ leads to

$$\mathscr{R}_{AC}(2,t) = \left(\frac{(3\lambda+\kappa+\mu)\sinh(\frac{t\sigma}{2})}{\sigma} + \cosh(\frac{t\sigma}{2})\right)e^{-\frac{(3\lambda+\kappa+\mu)t}{2}},$$

where

$$\sigma = \sqrt{\lambda^2 + 6\kappa\lambda + 6\mu\lambda + \kappa^2 + 2\kappa\mu + \mu^2}.$$

Similar systems of equations arise from the state-transition rate diagrams associated with Majority Consensus Voting and Dynamic-linear Voting.

3.2.2 Finite Spares

As shown in Figures 2 through 4, the state-transition rate diagrams for finite spares lead to more complex sets of equations. Each of these systems of linear, constant coefficient ordinary differential equations (ODEs) are of the form

$$P'(t) = AP(t)$$

with initial condition

$$P(0) = P_0$$

The solution is given analytically by

$$P(t) = e^{tA}P_0$$

where e^{tA} denotes the matrix exponential [19].

For simplicity of exposition, assume A has full geometric multiplicity. Its Jordan canonical form

$$A = T\Lambda T^{-1}$$

consists of the diagonal matrix Λ with eigenvalues, λ_i, of A on the diagonal and T, whose columns are the eigenvectors of A. The ODE can then be diagonalized:

$$P'(t) = T\Lambda T^{-1}P(t)$$

Defining

$$Z(t) = T^{-1}P(t)$$

the differential equation is

$$Z'(t) = \Lambda Z(t)$$

with solution

$$Z(t) = e^{t\Lambda}Z_0$$

where $e^{t\Lambda}$ is the diagonal matrix with entries $e^{t\lambda_i}, i = 1, ..., n$. The general solution is thus

$$P(t) = Te^{t\Lambda}T^{-1}P_0$$

which can be evaluated at any point t in time.

This procedure is costly. The vector $Z_0 = T^{-1}P_0$ need only be computed once, requiring approximately $\frac{1}{3}n^3$ flops, where a flop is a floating point add coupled with a floating point multiply. The n exponentials $e^{t\lambda_i}$ that comprise $e^{t\Lambda}$ would be formed for each value of t of interest at a cost which is linear in n. The total cost can be reduced by computing the solution at equally spaced points

$$t_k = k \cdot \Delta t$$

using

$$e^{t_{k+1}\lambda_i} = e^{\Delta t\lambda_i} \cdot e^{t_k\lambda_i}.$$

The propagation matrix

$$e^{\Delta t\Lambda} = \begin{pmatrix} e^{\Delta t\lambda_1} & 0 & \dots & 0 & 0 \\ 0 & e^{\Delta t\lambda_2} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & e^{\Delta t\lambda_{n-1}} & 0 \\ 0 & 0 & \dots & 0 & e^{\Delta t\lambda_n} \end{pmatrix}$$

need only be formed once, and later time step values can then be formed recursively from the previous step beginning with P_0 and using

$$P_{k+1} = T \cdot e^{\Delta t\Lambda} \cdot P_k$$

at a cost of one matrix-vector multiply (n^2 flops) per step. The major cost, though, arises when the eigensystem of A is computed.

Obtaining the eigensystem of A is equivalent to finding the roots of its characteristic polynomial. It was shown by Evariste Galois [11] that there is no direct method possible for computing the roots of an arbitrary polynomial of degree higher than four. This implies that models with five or more states will require an iterative process to obtain the eigensystem. The most effective is the QR algorithm [10]. Actual convergence of the iterative QR algorithm depends on the problem and the conditioning of the eigenvectors, but this one-time cost is estimated at $15n^3$ flops [10].

Thus, computing the matrix exponential directly is very costly and alternate methods of solution are desirable. Similar problems are typically attacked with general purpose ODE solvers such as RKF45 [8], based on the Runge-Kutta-Fehlberg method. This robust, reliable piece of mathematical software is capable of solving a general, nonlinear system of initial value problems. While it does not take advantage of the linear, constant coefficient nature of this particular problem, it is readily available and effective for small models. This system of ODEs imposes a fundamental limitation on any numerical solution since there are no transitions leaving the failed state. The resulting system of linear ODEs must have a zero eigenvalue. Because the remaining eigenvalues are all negative, the system is “infinitely stiff” [25].

A technique that does take advantage of the linear, constant coefficient system which characterizes our models of reliability is the Padé approximation [10]. The Padé approximation forms a rational polynomial approximation in which the numerator is a Taylor approximation to the matrix exponential

$$e^{\frac{t}{2}A} = I + \frac{t}{2}A + \frac{(\frac{t}{2}A)^2 2!}{+}\frac{(\frac{t}{2}A)^3}{3!} + \ldots$$

while the denominator is an inverse Taylor approximation to the matrix exponential

$$e^{\frac{t}{2}A} = \frac{1}{e^{\frac{-t}{2}A}} \approx \frac{1}{1 - \frac{t}{2}A + \frac{(\frac{t}{2}A)^2}{2!} + \frac{(\frac{t}{2}A)^3}{3!} + \frac{(\frac{t}{2}A)^4}{4!} + \frac{(\frac{t}{2}A)^5}{5!} + \ldots}.$$

Truncating each of these series after six terms yields an eleventh order Padé approximation to this propagation matrix. This approximation is only reliable near the origin, and therefore requires scaling [19]. The matrix ΔtA is scaled so that

$$\frac{\parallel \Delta tA \parallel}{2^j} \le \frac{1}{2}$$

The Padé approximation is formed for $e^{\frac{\Delta t}{2^j}A}$. The desired $e^{\Delta tA}$ can be recovered by repeated squaring of this n by n matrix j times. Since the Padé approximation series was terminated after six terms, the cost is $(j+5)n^3$ flops.

Some savings are possible for this particular problem. Since the eigenvalues are known to be real and negative, accuracy could have been delivered by Padé approximation series with fewer terms. The intent, though, was to have a general purpose code that would handle a wide variety of models. If some further models included positive eigenvalues, the six-term series would be required.

The scaling factor of 2^j depends on the frequency of output required for any desired graph. In Figure 6, a typical value required for j was six. The use of the Padé approximation is slightly less expensive than obtaining the eigensystem using the QR algorithm. Since the size of the ODE system is small, either technique could be preferred for this problem when output at equally spaced intervals is required.

3.2.3 Results

The eigenvalues provide additional information on the decay constants for the various models, and are useful in the initial analysis of the model. The nature of these systems admit solutions using the less expensive Padé approximation.

The probability of being in each of the states in Figure 2 as a function of time was calculated to observe the transitions between the different configurations of the Available Copy protocol with three sites and two spares. For the typical conditions $\lambda = 0.1, \mu = 1.0$, and $\kappa = 100$, all states from which regeneration can occur have neglible probability of being occupied due to the relatively large κ. Figure 6 illustrates the activity of the states with significant probability of being occupied.

To resolve the behavior near time zero, a scale of 10^{-2} for the grid spacing was needed for the initial transitions of the model. As the system evolved in time, the smoothing due to the decay caused by the negative, real eigenvalues allow a coarser grid spacing. The probability of being in a given state is plotted against the log of the independent time variable in Figure 7 using a log scale for the independent variable. This displays the details of the initial system behavior which occur on a small time scale, as well as the transition to steady state which occurs over a much larger scale.

The varying time scale in Figure 7 suggests using the eigensystem method. The Padé approximation provides an effective estimation of the propagation matrix for a fixed value of Δt at a lower cost than computing the matrix eigensystem. The semilog graph in Figure 7 employed time scales of $\Delta t = 10^{-2}, 10^{-1}, 10^{0}, \ldots, 10^{4}$, and 10^{5}. This required the computation of seven separate propagation matrices $e^{\Delta tA}$. Although the Padé approximation is more effective for the computation of a single propagation matrix, the eigensystem is more effective in approximating a family of propagation matrices for the same matrix A with different scales Δt.

For the semilog graphs, the eigensystem of A need only be computed once to obtain the eigenvalues $\lambda_1, \ldots, \lambda_n$ and the eigenvectors in the columns of the matrix T, and the results are applicable over all the time scales. Each value of Δt_j needed to resolve the scale of the independent variable in each particular model determines a propagation matrix $e^{\Delta t_j A} = Te^{\Lambda \Delta t_j}T^{-1}$. This requires the re-evaluation of the exponential terms $e^{\Delta t_j \lambda_i}$ which comprise $e^{\Lambda \Delta t_j}$, at a linear cost in n.

The Dynamic-linear Voting and Majority Consensus Voting models also have several highly improbable states. Though the behavior is qualitatively similar to that of Available Copy, evidenced by the same general shape of the curves, Figures 8 and 9 reveal a more rapid decay of the system. Dynamic-linear Voting performs better than static Majority Consensus primarily due to the increased probability of occupying state $\langle 3, 1 \rangle$.

To determine the relative reliability afforded by the protocols, the same three site, two spare system is analyzed for each protocol. Numerical solutions, validated with simulation results, were obtained for each of the protocols under identical conditions: $\lambda = 0.1, \mu = 1.0$, and $\kappa = 100$. In Figures 10 and 11, the discrete points reflect the deciles found by simulation, while the curves represent the numerical solutions found using the Padé approximation, displayed on a linear time scale.

Figure 10 illustrates the marked advantage of Available Copy over both quorum-based protocols, thus making it the protocol of choice in an environment in which network partitions are impossible. When partial communication failures can occur, Dynamic-linear Voting clearly surpasses Majority Consensus Voting. The relative ordering of the protocols agrees with the non-regenerative case [18], and further analysis shows that these relative advantages are independent of the number of sites and spares [15]. These conclusions are also independent of the rate of regeneration κ [15].

To examine the trade-off between storage space and reliability, the performance of the Available Copy protocol was compared for each reasonable allocation of sites and spares in a five-machine system. Figure 11 shows that replacing a single replica with a spare has only a slight detrimental effect on the reliability of the data object. Further decreasing the number of replicas markedly degrades the reliability. Similar limits are reached for the other protocols as well.

4 Conclusions

Replica control protocols that employ regeneration increase the reliability and availability of replicated data objects by creating new replicas when one or more of the replicas have been lost due to a system failure. A regeneration-based replica control protocol begins with a fixed number of replicas on various sites. As these replicas fail, they are replaced by new replicas that are created on additional sites called spares. Once a failed site has recovered, the regenerated replicas become superfluous and the additional storage used can be reclaimed.

There are costs associated with regeneration-based replica control protocols as well. Perhaps the most important is an increase in network message traffic. When a site fails, or the communication network becomes partitioned, a regeneration will occur. To generate a new replica, a copy of an extant replica must be transmitted. The cost of this may be prohibitive for large data objects such as data bases. For this reason, regeneration may be best suited for small data objects with strong fault tolerance requirements. Some examples of such objects are directories and mail boxes. It is particularly important for a directory to be available since no other data objects can be accessed without it.

There are several ways of mitigating the cost of regeneration [17]. One technique allows regeneration only when the number of replicas falls below a certain threshold. Another method delays regeneration for a period of time following the failure of either a site or the communications network. This has a dampening effect and prevents the protocol from wasting resources by reacting to transient failures. The performance of mitigated regeneration protocols should be evaluated to determine their effect on the reliability provided by the protocol. Further work is needed to assess the costs in terms of network message traffic resulting from regeneration, and to estimate the additional storage cost incurred.

With five or more participating sites and a high rate of regeneration, replacing a single replica with a spare has been shown to have only a slight detrimental effect on the reliability of the data object. As illustrated by Figure 11, the reliability decreases drastically if fewer replicas are maintained. Thresholds at which similar degradation occurs can also be observed for the other protocols that were considered.

Available copy protocols have been shown to provide the highest possible reliability for all replica control protocols studied, and these results parallel those obtained for availability [18]. Using standard Markovian assumptions, closed-form expressions have been derived for the reliability of the tractable systems. The simulations of systems employing sites with non-Markovian distributions show that the Markovian models upon which the numerical solutions are based are indeed robust enough to predict the behavior of non-idealized systems.

The numerical solutions of the Markov models, backed by simulation results, establish a hierarchy of systems ordered by increasing reliability. They clearly indicate that the Available Copy protocol provides much higher reliabilities than the quorum-based protocols, and establish Dynamic-linear Voting as the protocol of choice for a communications network susceptible to partitioning. With estimates of the mean times to site failure and repair, the numerical techniques presented here can be applied to predict the reliability afforded by differing apportionments of sites and spares. Designers may in this way determine the fewest number of replicas that can provide the desired level of reliability.

The three approaches employed in this paper, closed-form solutions supplemented with numerical solutions and validated by simulation results, have also been successfully applied to the replica control protocols that only repair existing sites [16, 18]. Similar analysis should be applied to any proposed protocol to determine its performance relative to existing techniques.

Acknowledgements

The authors are grateful to Jehan-François Pâris, Alexander Glockner and Mary Long for their assistance. The closed-form solutions were found with the aid of *Maple,* a symbolic algebra program developed by the Symbolic Computation Group at the University of Waterloo. The numerical results were computed using MATLAB from MathWorks, Incorporated of Sherborn, Massachusetts. The simulation results were obtained with the aid of SIMSCRIPT II.5, a simulation language developed and supported by CACI Products Company of La Jolla, California.

References

[1] P. A. Bernstein and N. Goodman. An Algorithm for Concurrency Control and Recovery in Replicated Distributed Databases. *ACM Transactions on Database Systems (TODS)*, 9(4):596–615, 1984.

[2] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley Publishing Company, 1987.

[3] J. L. Carroll and D. D. E. Long. The effect of failure and repair distributions on consistency protocols for replicated data objects. In *Proceedings of the 22th Annual Simulation Symposium (SS 1989)*, pp. 47–60, Tampa, Mar. 1989. IEEE.

[4] J. L. Carroll, D. D. E. Long, and J.-F. Pâris. Block-Level Consistency of Replicated Files. In *Proceedings of the Seventh International Conference on Distributed Computing Systems (ICDCS '87)*, pp. 146–153, Berlin, Sept. 1987. IEEE.

[5] J. L. Carroll, D. D. E. Long, and J.-F. Pâris. Block-Level Consistency of Replicated Files. In *Proceedings of the Seventh International Conference on Distributed Computing Systems (ICDCS '87)*, pp. 146–153, 1987.

[6] D. Davcev and W. A. Burkhard. Consistency and Recovery Control for Replicated Files. In *Proceedings of the Tenth ACM Symposium on Operating Systems Principles (SOSP '85)*, pp. 87–96, 1985.

[7] C. A. Ellis. Consistency and Correctness of Duplicate Database Systems. *ACM SIGOPS Operating Systems Review*, 11(5):67–84, 1977.

[8] G. E. Forsythe, M. A. Malcolm, and C. B. Moler. *Computer Methods for Mathematical Computations.* Englewood Cliffs: Prentice Hall, 1977.

[9] D. K. Gifford. Weighted Voting for Replicated Data. In *Proceedings of the Seventh ACM Symposium on Operating Systems Principles (SOSP)*, pp. 150–162. ACM, 1979.

[10] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, 1996.

[11] I. N. Herstein. *Topics in Algebra.* Blaidell Publishing Company, 1983.

[12] S. Jajodia and D. Mutchler. Dynamic voting. In *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, pp. 227–238. ACM Press, 1987.

[13] D. D. E. Long. *The Management of Replication in a Distributed System.* PhD thesis, Department of Computer Science and Engineering, University of California, San Diego, 1988.

[14] D. D. E. Long. On the Storage Requirements of Regeneration. Technical Report UCSC-CRL-89-04, University of California Santa Cruz, July 1989.

[15] D. D. E. Long, J. L. Carroll, and K. Stewart. The Reliability of Regeneration-Based Replica Control Protocols. In *Proceedings of the Ninth International Conference on Distributed Computing Systems (ICDCS '89)*, pp. 465–473, Newport Beach, June 1989. IEEE.

[16] D. D. E. Long and J.-F. Pâris. A Realistic Evaluation of Optimistic Dynamic Voting. In *Proceedings of the Seventh Symposium on Reliable Distributed Systems (SRDS '88)*, pp. 129–137, Columbus, Oct. 1988. IEEE.

[17] D. D. E. Long and J.-F. Pâris. Regeneration Protocols for Replicated Objects. In *Proceedings of the Fifth International Conference on Data Engineering (ICDE '89)*, pp. 538–545, Los Angeles, Feb. 1989. IEEE.

[18] D. D. E. Long, J.-F. Pâris, and J. L. Carroll. Reliability of Replicated Data Objects. In *Proceedings of the Eighth IEEE International Performance, Computing and Communications Conference (IPCCC '89)*, pp. 402–406, Phoenix, Mar. 1989. IEEE.

[19] C. B. Moler and C. F. V. Loan. *Nineteen dubious ways to compute the exponential of a matrix*. SIAM Review 20, 1978.

[20] J. Noe and A. Andreassian. Effectiveness of Replication in Distributed Computing Networks. In *Proceedings of the Seventh International Conference on Distributed Computing Systems*, pp. 508–513, 1987.

[21] J.-F. Pâris. Voting with Witnesses: A Constistency Scheme for Replicated Files. In *Proceedings of the Sixth International Conference on Distributed Computing Systems (ICDCS '86)*, pp. 606–612, 1986.

[22] J.-F. Pâris. Efficient Management of Replicated Data. In *Proceedings of the Second International Conference on Database Theory*, pp. 396–409. Springer Verlag, 1988.

[23] C. Pu. *Replication and Nested Transactions in the Eden Distributed System.* Ph.D. dissertation, University of Washington, 1986.

[24] C. Pu, J. D. Noe, and A. Proudfoot. Regeneration of Replicated Objects: A Technique and Its Eden Implementation. In *Proceedings of the Second International Conference on Data Engineering*, pp. 175–187. IEEE Computer Society, 1986.

[25] L. H. Shampine and C. W. Gear. *A user's view of solving stiff ordinary differential equations.* SIAM Review, 21, 1979.

[26] D. Skeen. A Quorum-Based Commit Protocol. In *Proceedings of the Sixth Berkeley Workshop on Distributed Data Management and Computer Networks*, pp. 69–80, Feb. 1982.

Figure Captions

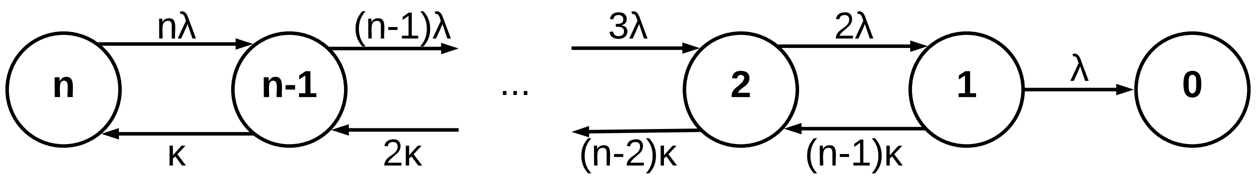

Figure 1: State Transition Diagram for Available Copy with *n* Sites and Unlimited Spares

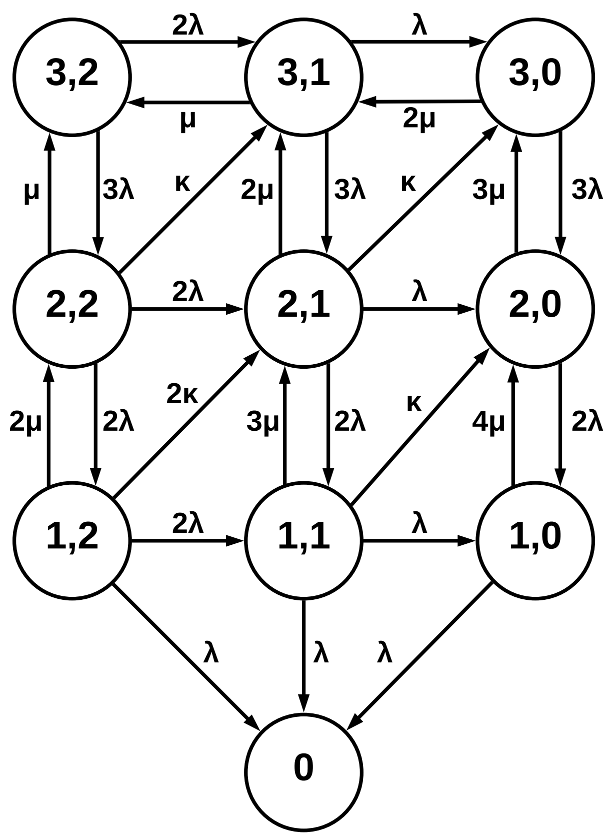

Figure 2: State Transition Diagram for Available Copy with 3 sites and 2 spares

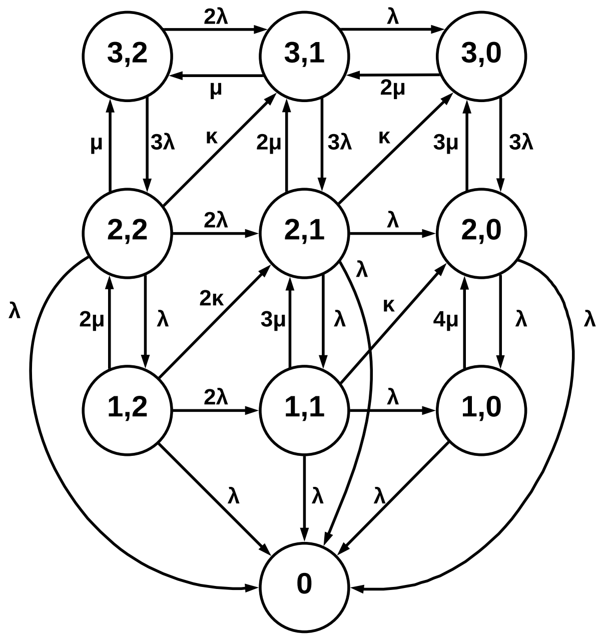

Figure 3: State Transition Diagram for Dynamic-linear Voting with 3 sites and 2 spares

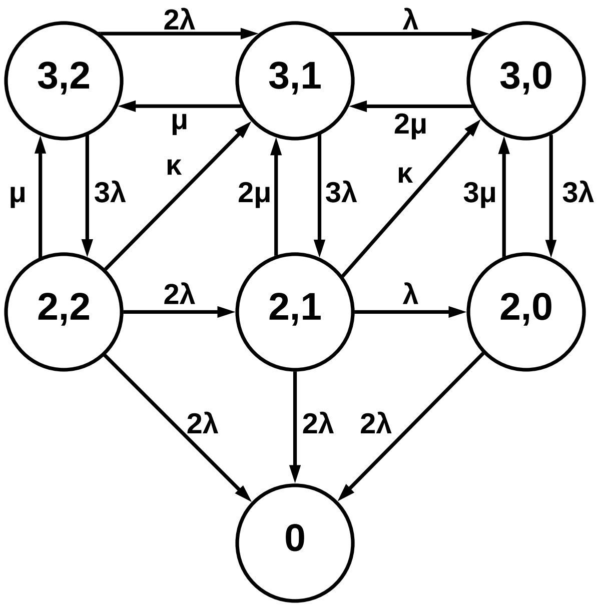

Figure 4: State Transition Diagram for Majority Consensus Voting with 3 sites and 2 spares

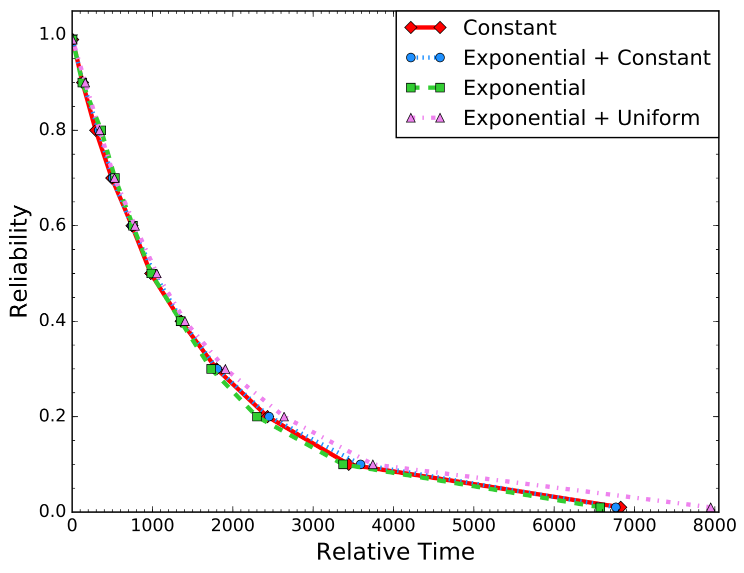

Figure 5: Compared Regeneration Distributions of 3 sites and 2 spares

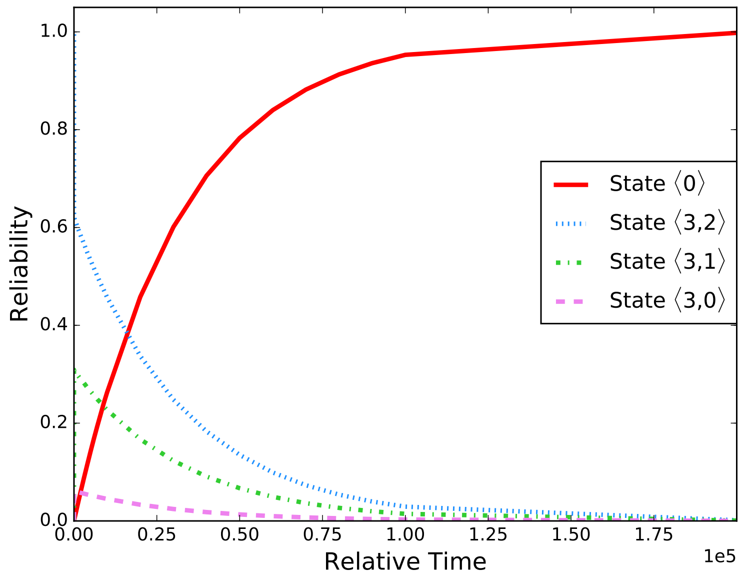

Figure 6: Available Copy with 3 Sites and 2 Spares: $\kappa = 100.0, \lambda = 0.1, \mu = 1.0$

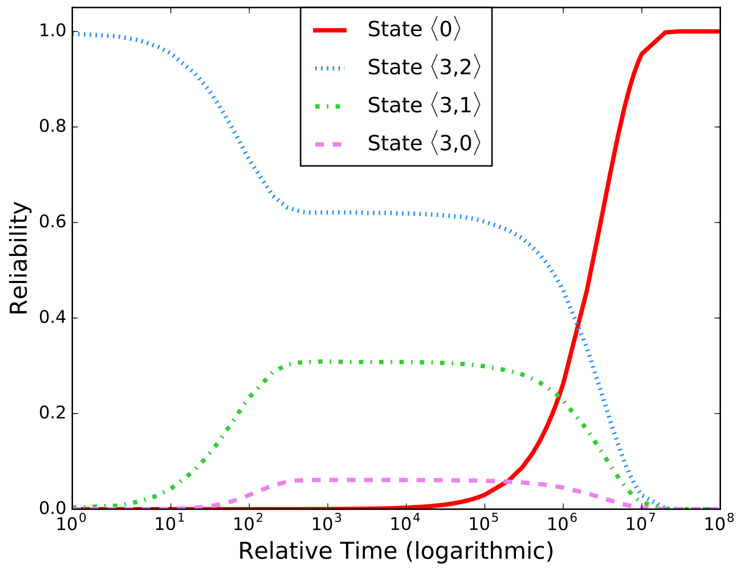

Figure 7: Available Copy with 3 Sites and 2 Spares: $\kappa = 100.0, \lambda = 0.1, \mu = 1.0$

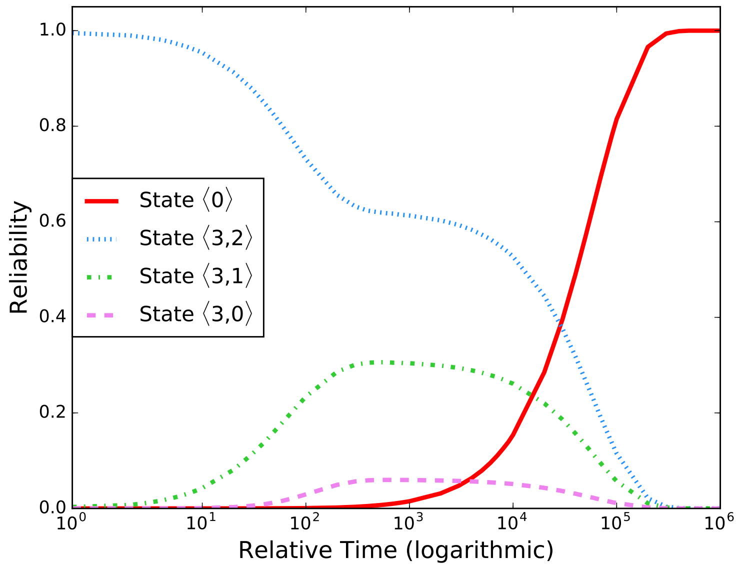

Figure 8: Majority Consensus Voting with 3 Sites and 2 Spares: $\kappa = 100.0, \lambda = 0.1, \mu = 1.0$

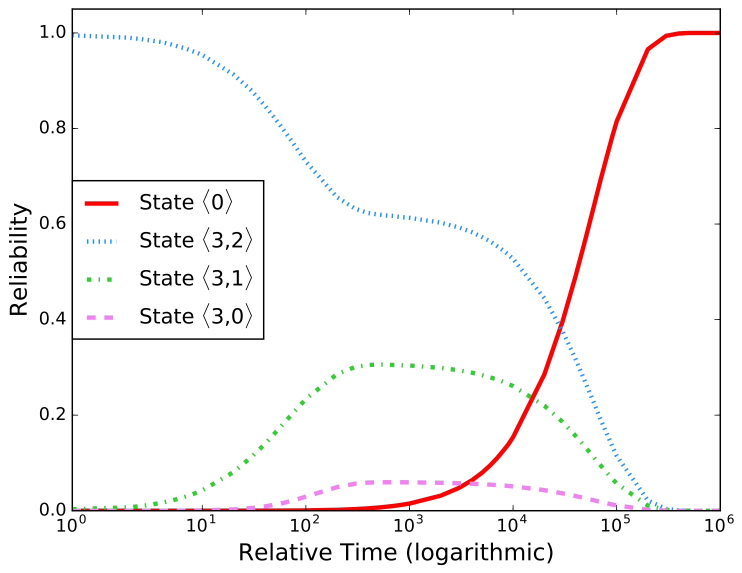

Figure 9: Dynamic-linear Voting with 3 Sites and 2 Spares: $\kappa = 100.0, \lambda = 0.1, \mu = 1.0$

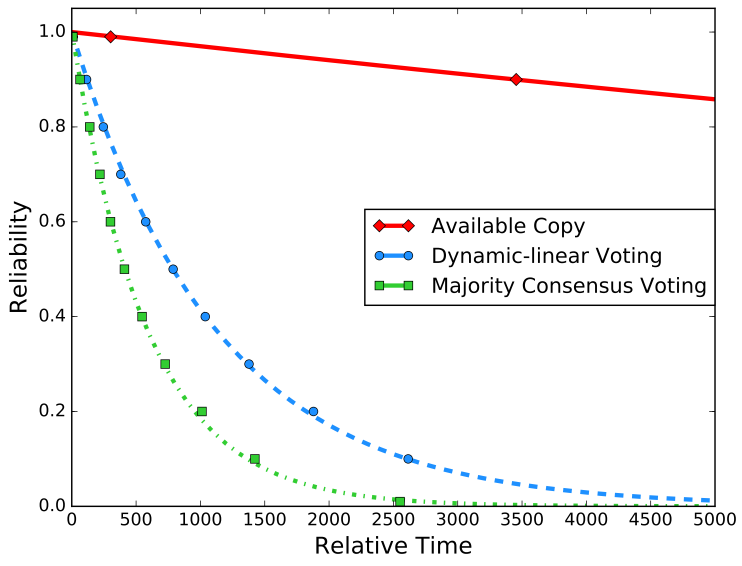

Figure 10: Compared Reliability of AC, DLV and MCV: $\kappa = 100.0, \lambda = 0.1, \mu = 1.0$

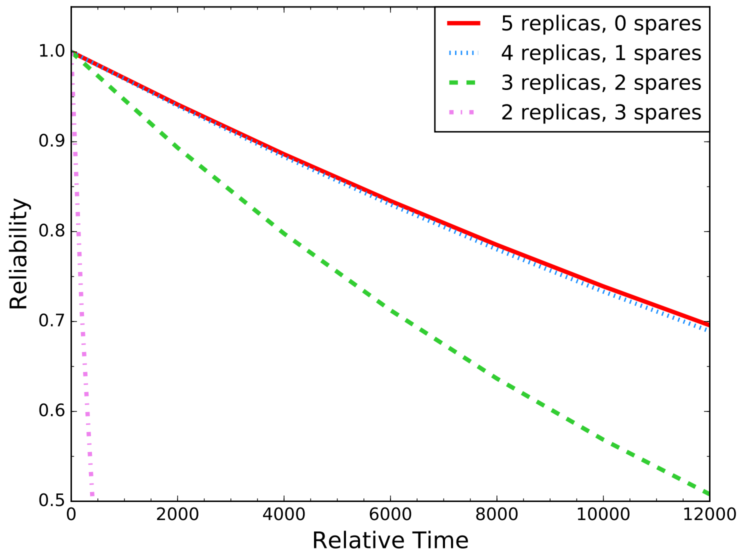

Figure 11: Compared Reliability for AC with Varying Numbers of Spares: $\kappa = 10.0, \lambda = 0.1, \mu = 1.0$