Using Simulation to Evaluate Consistency Protocols for Replicated Data

Darrell D. E. Long
University of California, Santa Cruz
Santa Cruz, CA 95064
408-429-2616

John L. Carroll
San Diego State University
San Diego, CA 92115
619-594-7242

May 28, 2003

1 Introduction

In a distributed system, data are often replicated for protection against site failures and network partitions. When data are replicated at several sites, a consistency control policy must be chosen to ensure a consistent view of the data and maintain the appearance that there is only a single replica of the data. The view presented to the user must remain consistent even in the presence of site failures and network partitions. Sites recovering from a failure must present the data stored at that site in such a way that it is consistent with the global view of the data. Several consistency control protocols have been proposed, and the most promising are variants of available copy schemes and schemes based on quorum consensus.

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 degree of fault tolerance achieved depends on both the degree of data redundancy and on the consistency protocol that is used to manage the object. 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.

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

Simulation plays a crucial role in evaluating the performance of these protocols. Consistency control protocols limit access to the data object to ensure that only current copies of the data can be accessed. The effect these protocols have on the accessibility of the replicated data is investigated by simulating the operation of the network and measuring the performance. Several strategies for replica maintenance are considered, and the benefits of each are analyzed using simulators which have been validated by numeric solutions derived from Markov models. The details of the simulations are discussed. Measurements of the reliability and the availability of the replicated data are compared and contrasted. In particular, the sensitivity of the protocols to common patterns of site failures and repairs is studied in detail, and the effect the second moments have on the results is analyzed. The relative performance of competing protocols is shown to be only marginally affected by non-exponential distributions, validating the robustness of the exponential approximations.

Stochastic process models have been used extensively to evaluate the fault tolerance provided by replica control protocols. However, there are many issues that cannot be addressed by stochastic process modeling. The assumption of an exponential distribution for site repair times is unrealistic, but using other distributions result in intractable analytic models. The problem of modeling partitions of the communication network is intractable for all but the most basic cases [34], since the number of states that must be considered quickly becomes unmanageable. These are among the reasons that discrete event simulation has been chosen to evaluate the fault tolerance provided by replica control protocols.

Discrete event simulation provides the ability to model replicated data objects in a more realistic manner. It allows for the relaxation of many of the assumptions that are necessary for an analytic

model of the replicated data object. An added advantage of using simulation is that it can easily model systems composed of many network segments. The primary disadvantage of discrete event simulation is that the solutions obtained pertain only to the specific configuration being modeled. This makes it difficult to predict how the system will behave when some of the parameters are modified.

In §2, the consistency control protocols evaluated in this study are introduced. The regeneration enhancements to these protocols are described in §2.3.1. In §3, the problem of evaluating the performance of replica control protocols is considered. The task of accurately determining simulation parameters poses some interesting problems and is discussed in §3.1. A simulation for measuring the reliability afforded by the various replica control protocols is presented in §4, and the simulation model for measuring the availability provided by these is discussed in §5. The results obtained from the availability study are presented in §5.3, followed by the conclusions in §6.

2 Replica Control Protocols

The field of consistency control protocols for replicated data objects has existed for about ten years. Its birth coincides with the advent of distributed data bases and the network technology required to support them. When data objects are replicated around a computer network a protocol must be chosen to ensure a consistent view to an accessing process. The replicas of the data object are then said to be mutually consistent. The protocols used to ensure mutual consistency are known as *replica control* or *consistency control* protocols.

A replicated data object has the same semantics as an unreplicated instance of the same data object. The accessibility 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 fault tolerance 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 major replica control protocols are presented next, followed by a discussion of the

modifications necessary to incorporate regeneration in each scheme.

2.1 Available Copy

The *Available Copy* protocol, is a descendent of the *Write All Available* protocol used in the SDD-1 distributed data base system [4, 5]. It is comprised of a set of policies for providing mutual consistency and concurrency control in a distributed data base system. The protocol allows reads and writes of the data object until all sites are simultaneously in a failed state. After such a total failure, recovering sites must reliably determine whether they hold the most recent version of the data object. Access to the data object is restored only after the last site to fail has been identified; when this site recovers, it is used to determine the current state of the data object.

The Available Copy protocol requires that some very strong assumptions be made about the operating environment. The first assumption is that the communication network must be reliable. The communication network must not be susceptible to partitions, and reliable message delivery must be assured. A more controversial assumption is that failures are detectable in a fool-proof manner. In practice this is often a very difficult task. The method that is used is to send a message to a site and wait for a response or for it to time-out. This is unreliable in the case of heavily loaded sites.

Due to the strong assumptions that are made about the communication network, the Available Copy protocol guarantees that all available replicas of the data object are current. By assuming that the network will provide reliable delivery of each message mutual consistency can be ensured by sending each write to every available copy. This also allows reads to be performed at any available copy, in particular, at the local site if a copy is present.

When a site recovers from a failure it must obtain a current replica of the data object. If there is another operational site that holds a current replica then the recovering site can request a replica of the data object from that site. In the event of a total system failure the Available Copy protocol must determine the last site to fail since that site must hold a current replica of the data object. The problem of determining the last site to fail has been studied by Skeen [32].

In the original proposal for the Available Copy protocol [2], the goal of detecting the last site to

fail was achieved by maintaining several sets of failure information, including: all sites participating in the replication of the data object, those sites that have been specifically included in the set of replicas, and those sites that have been specifically excluded from the set of replicas. An *included* site is one that is known to hold a replica of the current version of the data object, and an *excluded* site is one that has failed and the failure has been detected by some other operational site.

2.2 Quorum Consensus

Quorum consensus protocols ensure the consistency of replicated data objects by honoring read and write requests only when an appropriate quorum of the replicas of the data object can be accessed [11, 33].

There are several methods for achieving this goal. The simplest is to assign a single vote to each site participating in the replication of the data object. When an access occurs each operational site sends its vote to an access coordinator managed by the protocol, which then determines if a quorum is present. This can be refined by assigning a variable number of votes to each site depending on its characteristics.

Other alternatives have also been proposed. Different quorums can be assigned for read and write operations allowing the protocol to be tailored to the most frequent type of access. The number of votes assigned to each site, or even the required quorum, can be adjusted in response to changing system conditions [1, 10]. This analysis is confined to an egalitarian model based on one vote per site.

2.2.1 Static Quorum Consensus

In their simplest form quorum consensus protocols assume that the correct state of a replicated data object is the state of the majority of its replicas. This simple static case is known as *static Majority Consensus Voting.* Ascertaining the state of a replicated data object requires collecting a quorum of the replicas. Should this be prevented by one or more site failures, the replicated data object is considered to be unavailable.

Simple quorum consensus protocols can be refined by introducing different quorums for read and write operations or by allocating different weights, including none, to each replica [11]. Mutual

consistency among replicas of the data object is guaranteed as long as the write quorum is high enough to disallow simultaneous writes on two disjoint subsets of the replicas, and the read quorum is high enough to disallow simultaneous reads and writes on two disjoint subsets of the replicas. These conditions are simple to verify and account for much of the conceptual simplicity and the robustness of quorum consensus protocols.

Although static vote assignments are simple to understand and to implement, they do not provide the highest possible level of fault tolerance. Dynamic quorum adjustment improves the performance of quorum consensus by responding to changes in the state of the system.

2.2.2 Dynamic Quorum Adjustment

Dynamic quorum adjustment protocols adjust the required quorum of replicas in response to changing system configurations. Among these are *Dynamic Voting* [7], *Dynamic-linear Voting* [12] and *Optimistic Dynamic Voting* [20].

A weakness of all static quorum consensus protocols is that the quorum is fixed and does not change once the system has begun operation. Because of this the loss of one half of the votes will render the data inaccessible. Davčev and Burkhard [7] proposed a solution to this problem, known as *Dynamic Voting.* Their policy adjusts the quorum of replicas required for an access operation automatically. A group of replicas, comprised of a majority of the current replicas that can communicate among themselves, is referred to as the *majority partition.*

The original Davčev-Burkhard Dynamic Voting protocol is based on the *connection vector.* The connection vector instantaneously records the state of the system with respect to all sites. Each replica of a data object has an associated ensemble of state information consisting of the version number and the partition vector. The *version number* of a replica represents the number of successful write operations to the replica. The *partition vector* at a site records the version numbers of all sites with respect to that site. The system state information is assumed to propagate instantaneously to all sites in the computer network.

In its original form, Dynamic Voting allows accesses to proceed so long as a strict majority of the *current* replicas are accessible. In situations where the number of current replicas within a

group of mutually communicating sites is equal to the number of missing replicas, Dynamic Voting cannot proceed and declares that the replicated data object to be inaccessible. A simple extension proposed by Jajodia [12], known as *Dynamic-linear,* enhances Dynamic Voting by resolving ties by applying a total ordering to the sites. The sites holding replicas of the data are given a static linear ordering. When a tie occurs, if the group of communicating sites contains exactly one-half the current replicas and contains the maximum element among the group of current replicas, that group is declared to be the majority partition.

A second protocol proposed by Jajodia and Mutchler [13], which they call *Dynamic Voting,* operates using only the number of sites that participated in the last write operation. Although read operations are known to be much more frequent than writes [23], no information about the system configuration is exchanged when a read operation occurs. As a result, the level of fault tolerance that the protocol can provide is decreased.

A third protocol proposed by Jajodia and Mutchler [14], combines Jajodia's total ordering with his previously proposed Dynamic Voting protocol. This protocol, now known as *Dynamic-linear Voting,* uses a designated site when an even number of replicas are present in order to break ties.

All Dynamic Voting protocols require that a quorum be collected for either a read or a write operation to be performed. The Optimistic Dynamic Voting protocol [20] takes advantage of this opportunity to adjust its view of the system configuration, thus improving the performance of the protocol. The view at each site consists of the *names* of each site to participate in the last successful operation with that site, and *operation number* and a *version number* of the data object. The operation number is incremented each time a successful operation occurs and represents a version number of the *partition sets* containing the names of the sites in the majority partition.

2.3 Regeneration

Pu [28] first proposed the Regeneration Algorithm as a space-efficient technique for increasing the availability of replicated data objects in the *Eden* system [29]. His protocol provides mutual and serial consistency of replicated data objects in a partition-free distributed system. 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. Noe and Andreassian suggested the adoption of an approach similar to the Available Copy protocol [2, 3] to alleviate the problem of poor reliability for writes. This approach allows writes to occur as long as a single replica of the data object remains available [22]. 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.

The hybrid protocols presented in this section successfully address the limitations of the original Regeneration Algorithm. When the communications network is not susceptible to partitioning, an Available Copy protocol [2, 3] provides the best fault-tolerance characteristics. When communications failures can occur, a consensus-based protocol based on Dynamic Voting [7] 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 [13] and Optimistic Dynamic Voting [20]. This technique can also be applied to static Majority Consensus Voting [8, 11].

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 [17] for typical failure and repair rates.

2.3.1 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, where to avoid possible inconsistencies it is essential to disenfranchise replicas that belong to previous generations.

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 [31] is used to ensure atomicity.

2.3.2 Regenerative Available Copy

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 yields better performance than the Regeneration Algorithm, which disallows writes when less than a full complement of sites are available.

The Optimistic Available Copy protocol [6, 19] provides all the necessary facilities to integrate regeneration into an available copy protocol. 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 contains the names of those sites which have the most recent version 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 most favorable reliability characteristics, but factors such as communication costs may make other choices more appropriate.

2.3.3 Regenerative Majority Consensus

The notion of generations is required to combine regeneration with static Majority Consensus Voting. 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* [26].

Adding regeneration is a simple extension to static Majority Consensus Voting [8, 11]. 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.

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.

2.3.4 Regenerative Dynamic Voting

The Optimistic Dynamic Voting protocol [20] is an implementation of Dynamic Voting [7], similar to Dynamic-linear Voting [13] 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 regener-

ation 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.

3 Modeling Considerations

The performance of many consistency protocols has been extensively evaluated [6, 13, 19, 20, 25, 24], but most of these studies are predicated on simplified assumptions about the site characteristics. Analytic solutions to performance measures such as availability and reliability are intractable unless the system is a homogeneous network of sites conforming to standard Markovian conditions. Such studies do not apply to distributed systems comprised of heterogeneous sites, and discount many inescapable events such as periodic down times required for scheduled maintenance.

The behavior of Markov processes is *robust,* that is, Markov processes are good approximations to the behavior of more general stochastic processes. It is important to judge the extent to which the theoretical performance of replicated data on idealized systems agrees with the observed performance of actual systems. The infrequency of site failures in computer systems precludes the compilation of accurate estimates of the failure and repair distributions, and hence simulation is an appropriate vehicle for estimating the robustness of various protocols.

The investigation of the effects that non-exponential distributions have on the fault tolerance characteristics of several consistency protocols is one goal of this study. Toward that end, simulation models are developed for some common performance measures. These models can also be used to collect other performance data, such as the mean time to repair and mean time to failure of the replicated data object. By employing exponential distributions on identical sites, the simulation models can be validated against the results predicted by closed-form and numerical solutions [6,

25, 24, 18].

The simulation models are used to investigate the sensitivity of the performance measures to the higher moments of the failure and repair distributions. Exponential, Erlang, uniform, and hyperexponential distributions are used, and the effect the second moments have on the results is considered. The degree to which the relative performance of competing protocols is affected by non-exponential distributions is presented. Reliability is explored in §4, and §5 includes a similar analysis of availability. The availability and reliability results are also compared with those from a simulation based on site failure data collected from functioning networks over a period of eight months.

In order to perform an analytic study of replica control protocols, a simplified system model must be developed. The replicas of the data object are assumed to reside on distinct sites of a computer network. 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 and does not fail. For this simplified model, 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. When a site fails, a repair process is immediately initiated at that site. Should several sites fail, the repair process will be performed in parallel on those sites. Site failures are assumed to be exponentially distributed with mean rate λ. The ratio of λ to μ is denoted by ρ; smaller values of ρ yield more reliable systems. 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. The resulting system is characterized by a discrete-state Markov process [34]. These assumptions preclude network partitions, which can cause several sites to become simultaneously inaccessible, and clearly apply only if the participating sites have independent power sources.

Such stochastic process models have been used extensively to evaluate the fault tolerance provided by replica control protocols. However, there are many issues that cannot be addressed by stochastic process modeling. The assumption of an exponential distribution for site repair times

is unrealistic, but using other distributions results in intractable analytic models. Discrete event simulation provides the ability to model replicated data objects in a more realistic manner. It allows for the relaxation of many of the assumptions that are necessary for an analytic model of the replicated data object.

3.1 Parameter Estimation

Estimating the parameters for the simulation models was found to be a difficult task. The first attempt involved asking system administrators how often their machines failed. From this experience it was learned that system administrators are hopelessly optimistic, and that their usual response is "almost never."

System administrators are more helpful when it comes to determining the service time for a component failure. The service contract for each machine specifies the period of time during which the service technician must arrive. By observing service technicians at work, a good estimate of how long it takes to diagnose the failure can be gained.

Some simplifying assumptions were necessary for the simulation model. The first is that the service technician will arrive within a known and bounded period of time. This is justified by the usual service contract clause that requires a service technician to arrive within a specified interval. The service technician is assumed to require a variable period of time to diagnose the failure. Once the technician has diagnosed the problem, the failed component must be replaced. In some cases the component will not be available and will require the technician to retrieve it from the warehouse. For simplicity, the portion of the service time during which the machine is under repair is assumed to be exponentially distributed.

The Berkeley Unix system provides a convenient method of collecting these statistics. Each site periodically sends a broadcast packet that lists the number of users, the load average and the time that the system has been operational. Since sites tend to be very reliable, it was impossible to gather enough data points during the lifetime of most machines to accurately determine the failure and repair distributions. The statistics gathering-processes were placed at several installations, including the University of California, San Diego and the Naval Ocean Systems Center. The Naval

Ocean Systems Center results were the most accurate since the system administrator there agreed to install them into the system restart script.

Several items of interest were encountered during the course of this effort. The first is that systems fail much more often than system administrators would lead the users to believe. This is not a conscious deceit on their part; it just seems that they do not regard "a quick reboot" as a system failure. Another is that significant periods of time can elapse between when a system fails and when it is restarted. The fluctuations in usage during different periods is a significant factor. The time to restart the system seems to be dominated more by the use of the machine than by the machine type. If a machine is in constant use, it is more likely to be rebooted quickly than one that is seldom used.

4 Reliability Analysis

Reliability is a measure of the probability that a protocol will continuously allow access to the replicated data object over a given time interval. With sufficient iterations, a reliability simulator can build a comprehensive picture of the time-dependent characteristics of a protocol.

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. 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.

Definition 4.1 *The* reliability $\mathcal{R}_P(n,t)$ *of an n-site system managed by protocol P is defined as the probability that the system will operate correctly over a time interval of duration t given that all n units were operating correctly at time* $t = 0$ [34].

The differential equations describing the behavior of systems managed by the replica control protocols can be derived from the state-transition flow rate diagrams. These equations only apply if all the distributions are exponential, and thus while they do not yield exact solutions for realistic scenarios, the solutions can validate the simulation model. In a network with a large cluster of sites, 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 states in the Markov chain which models such a system 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 $1 \leq i \leq n$, 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. The inaccessible replicas are on failed sites or sites still recovering from failure. 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.

Figure 1: Available Copy with Unlimited Spares

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. For example, the flow rate diagrams for three sites with two spares managed by Majority Consensus Voting is shown in Figure 2.

Figure 2: Majority Consensus with 3 Copies and 2 Spares

For small numbers of sites, closed-form solutions for the reliability of some of the protocols can be obtained from the differential-difference equations derived from the flow rate diagrams. 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. The comparison of the exponential and nonexponential models requires simulation, since the analytic analysis of the nonexponential systems is intractable.

4.1 Reliability Model

The availability simulation model and the reliability simulation model conform to the same basic guidelines. However, reliability can be measured by simulating the system just to the point of the first access failure. Since the availability simulator must also model the recovery mechanisms, the reliability simulator is comparatively simple. A complete listing of the latter is given in the appendix.

This simulation is programmed in SIMSCRIPT II.5 and executed on an ELXSI 6400 computer. Each site is modeled as a SIMSCRIPT process which is initially designated as being in an operational state. As time progresses, these processes alternate between operational and failed states according to the failure and repair distributions attributed to the corresponding site. When a site fails, its process is responsible for consulting the requirements of the protocols to determine whether the replicated data object can still be accessed from other sites; the first access denial of each protocol is recorded. In this fashion, the behavior of several competing protocols for given failure and repair distributions can be analyzed in a single simulation. This is not only efficient, but leads to more accurate comparisons since each protocol will be subject to an identical sequence of machine disruptions. An iteration terminates after all the protocols would deny access to the data object, that is, when all sites are simultaneously in a failed state.

The reliability graphs are the result of simulating the repairs and failures of a system of n sites until all sites failed, and noting the time at which each protocol would first deny access to a replicated data object. The process is repeated 1,000 times, and the results sorted to obtain an approximation of the reliability function.

4.2 Results of the Study

The degree to which analytic models predict the behavior of actual systems is of great practical concern. As discussed in §3.1, determining the site characteristics proved to be a difficult task. To investigate this relationship, nine sites were chosen from the data collected at the Naval Ocean Systems Center over a period of eight months [16]. The simulation uses these data to estimate the reliability of the sites. As shown in Figure 3, the behavior of the system based on this observed

data is satisfactorily approximated by the exponential models.

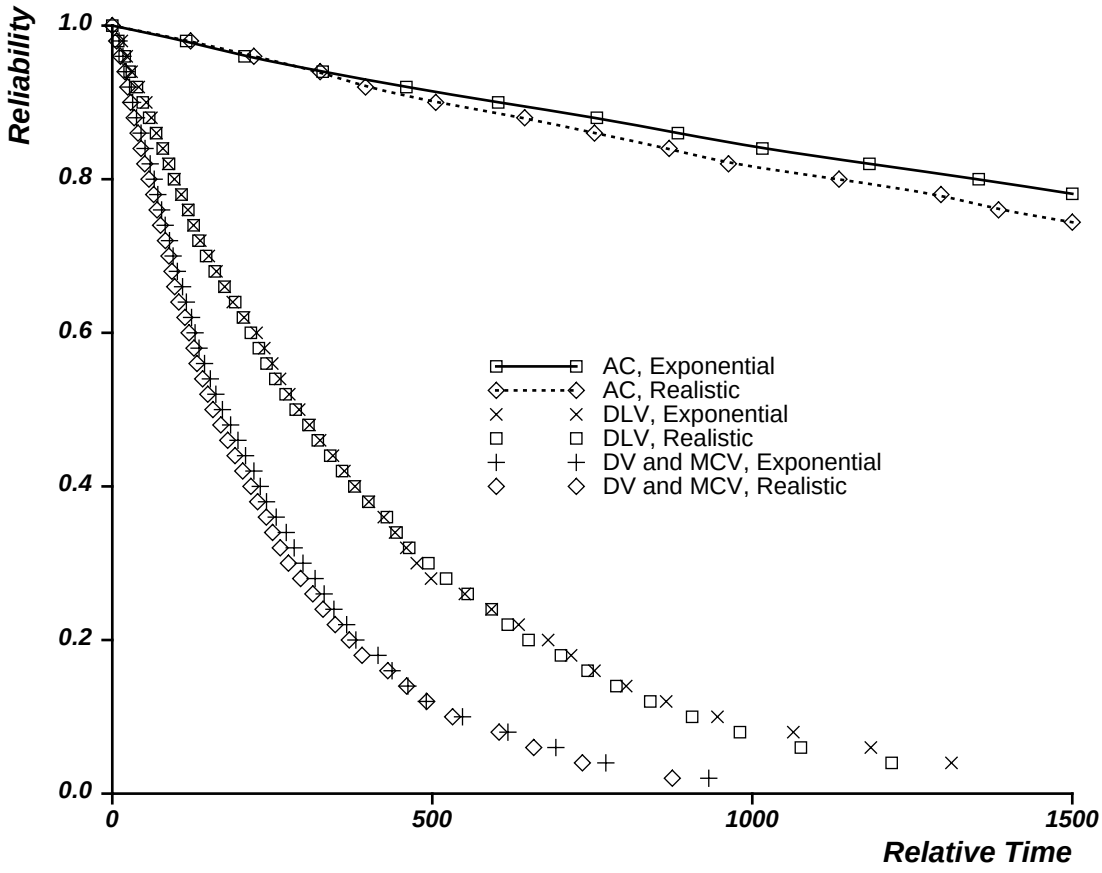

Figure 3: Reliabilities with three copies, Exponential versus Realistic Distributions

Figure 3 also illustrates the performance differences of the major consistency control protocols. Available Copy (AC) clearly has vastly superior reliability in comparison to the other protocols. Indeed, it has been shown to have the highest reliability that can be achieved for a given number of sites [21]. Among the protocols that can ensure consistency in the presence of network partitions, Dynamic-linear Voting (DLV) outperforms the other voting protocols.

Figure 3 reflects the relative performance for $n = 3$ sites, but the conclusions presented here also hold for larger networks of sites. For three copies, the superiority of Dynamic Voting (DV) over Majority Consensus Voting (MCV) is lost. The horizontal time axis is measured in units which correspond to the average time to perform a single site repair. In each of the following reliability graphs, ρ is fixed at 0.1, a typical value for modern systems. The following exposition focuses

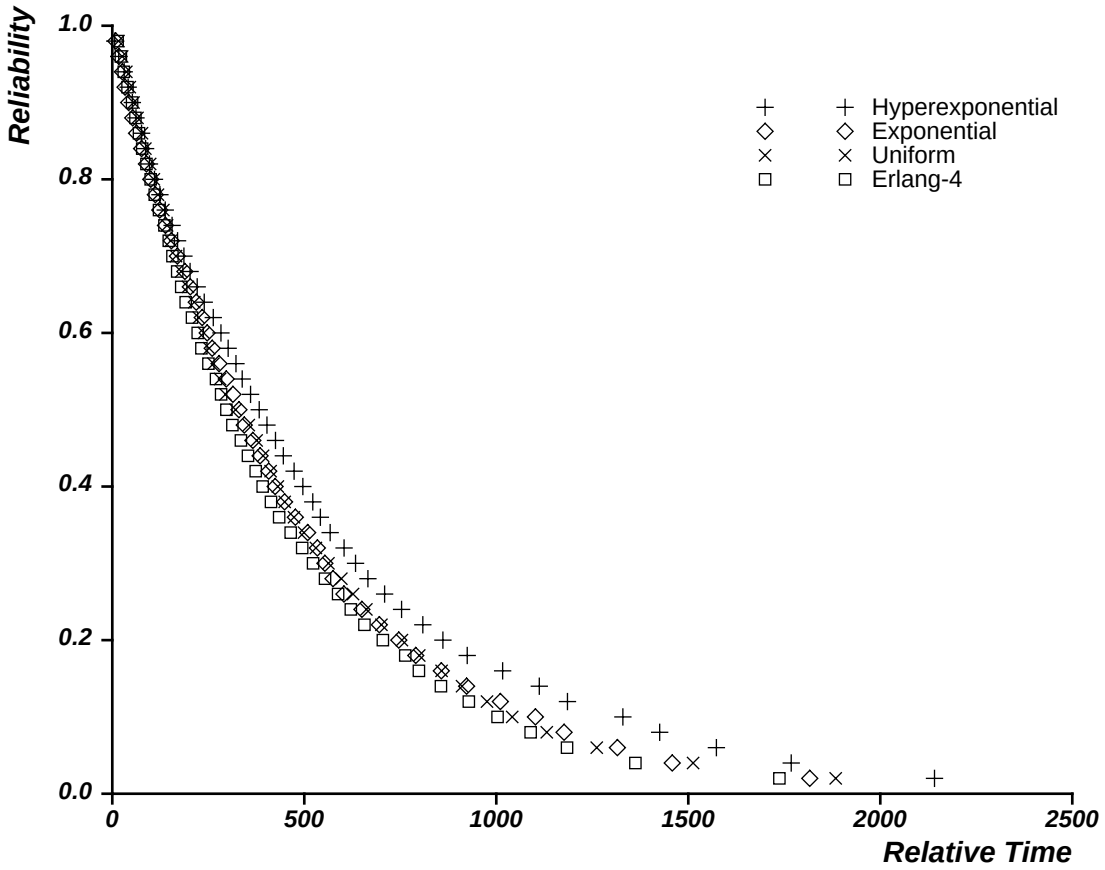

Figure 4: Reliability of three sites managed by Available Copy

on Available Copy and Dynamic-linear Voting, the protocols of choice in non-partitionable and partitionable networks, respectively.

Four representative distributions are investigated here and in §5 to determine the robustness of the theoretical solutions corresponding to exponential distributions. The performance resulting from exponential distributions is contrasted with that of Erlang-4, uniform, and hyperexponential distributions. The shape of each repair distribution is similar to the shape of the corresponding failure distribution; mixing the types of distributions within a single simulation did not enhance the analysis. For similar reasons, only homogeneous networks are considered for validation purposes: all sites are assigned identical characteristics.

In exponential distributions, the mean and standard deviation are equal, yielding a "moment ratio" of 1. The standard deviation of an Erlang-4 distribution is just half the mean: 0.5 is the

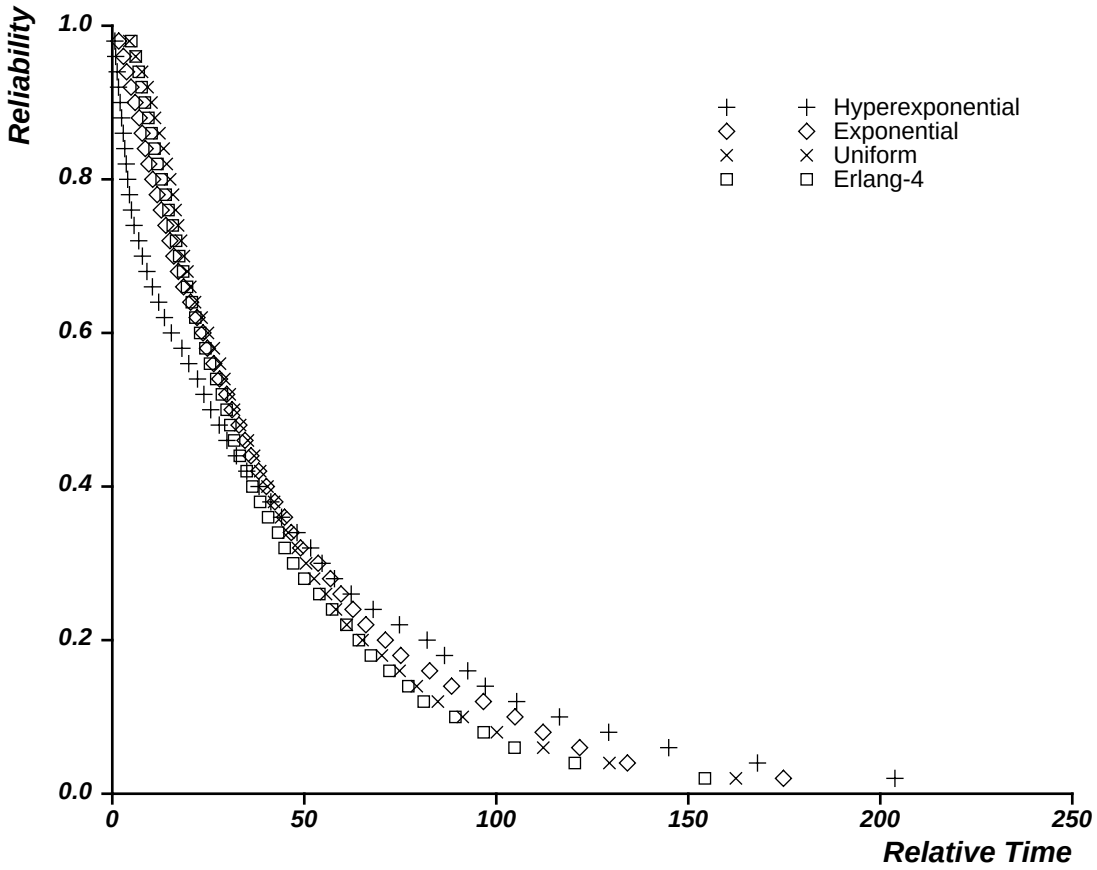

Figure 5: Reliability of three sites managed by Dynamic-linear Voting

smallest moment ratio considered in this study. The uniform distribution extends from 0 to twice the mean, and has a moment ratio of about 0.58. The ratio of the first and second moments in the hyperexponential distribution is approximately 1.5, the largest in this study. This is achieved by blending two exponentials, one with a mean nine times larger than the other.

Figures 4 and 5 illustrate the simulated reliability of the Available Copy and Dynamic-linear protocols, respectively. The four sample distributions have identical first moments but disparate higher moments. The differing times scales in the two figures reflect the much higher reliability of Available Copy.

The graphs show that the reliability of these systems is relatively insensitive to the shape of the distributions. As a rule, the uniform distribution has the poorest reliability, while the hyperexponential distribution has the highest reliability. These are the distributions with the

smallest and largest second moments, respectively.

For large networks, a larger moment ratio seems to ensures higher reliability, as illustrated in Figure 4. The correlation between the moment ratio and the reliability is somewhat weaker in the Dynamic-linear simulation, since it is possible to fail if only two of the three sites are down. In particular, the construction of the hyperexponential from two disparate exponentials implies that there will be both a larger number of simulation runs that experience quick failures, and a few instances in which the data object remains accessible for an unusually long period. The crossing of the hyperexponential curve in Figure 5 reflects this behavior. These effects are mitigated when a large number of sites are required for failure.

Each of the above experiments reflect a failure-to-repair ratio ρ of 0.1. It can be shown that a small increase in the reliability of the individual sites leads to an impressive increase in the reliability of the replicated object. However, the relative performance of the various protocols are unchanged, even when heterogeneous sites with varied distributions and mean failure and repair rates are simulated.

4.2.1 Regeneration Results

To determine the relative reliability afforded by the protocols, the same three site, two spare system is analyzed for each protocol. Both numerical solutions and simulation results were obtained for each of the protocols under identical conditions: $\lambda = 0.1, \mu = 1.0$, and $\kappa = 100$. In Figures 6 and 7, the discrete points reflect the deciles found by simulation, while the curves represent the Padé approximation of the numerical solutions to the differential-difference equations derived from the flow rate diagrams. These figures illustrate that when the simulation employs exponential distributions, the two methods yield strikingly similar results, effectively validating the models [18].

Figure 6 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

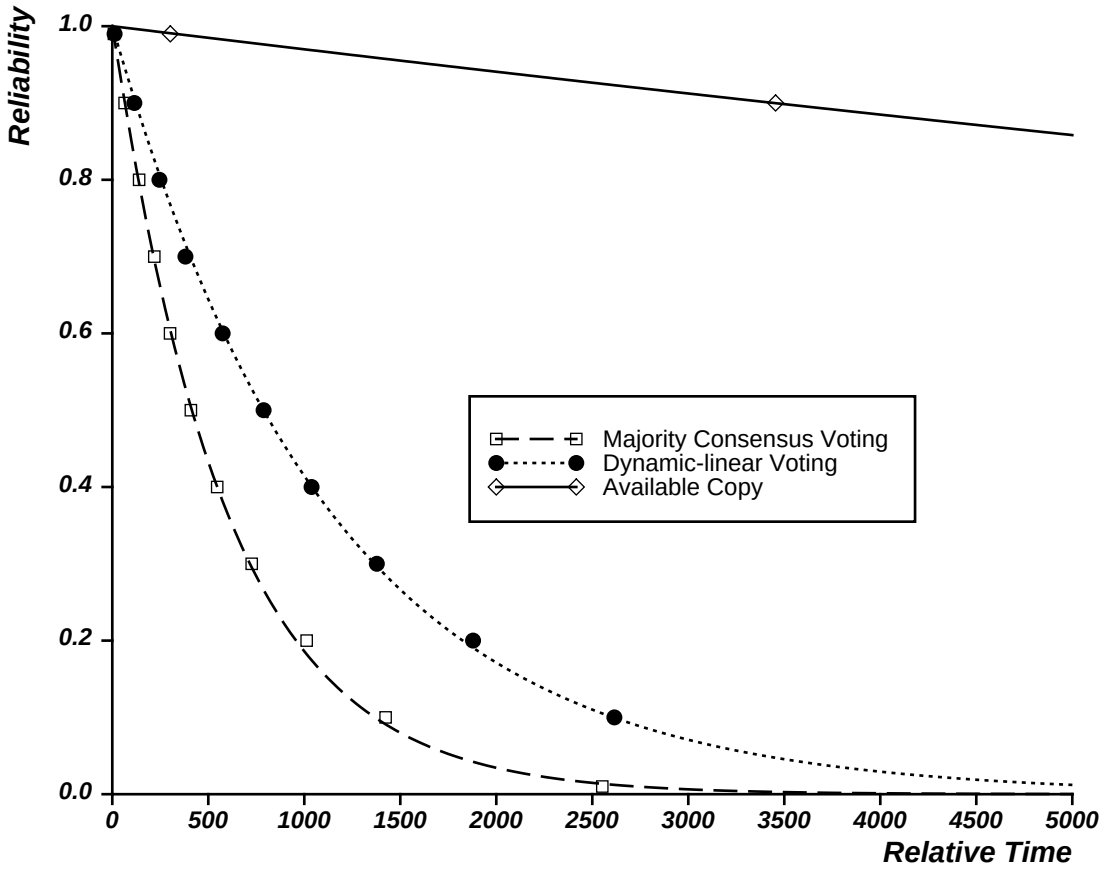

Figure 6: Compared Reliability of AC, DLV and MCV: $\kappa = 100.0, \lambda = 0.1, \mu = 1.0$

[21], and further analysis shows that these relative advantages are independent of the number of sites and spares [18]. These conclusions are also independent of the rate of regeneration κ [18].

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. Figure 7 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. Thresholds at which similar degradation occurs can also be observed for the other protocols considered.

5 Availability Analysis

While reliability gauges the transitory dynamics of a system, availability measures its steady-state behavior. The analysis is complicated by the necessity of modeling the recovery mechanism of the

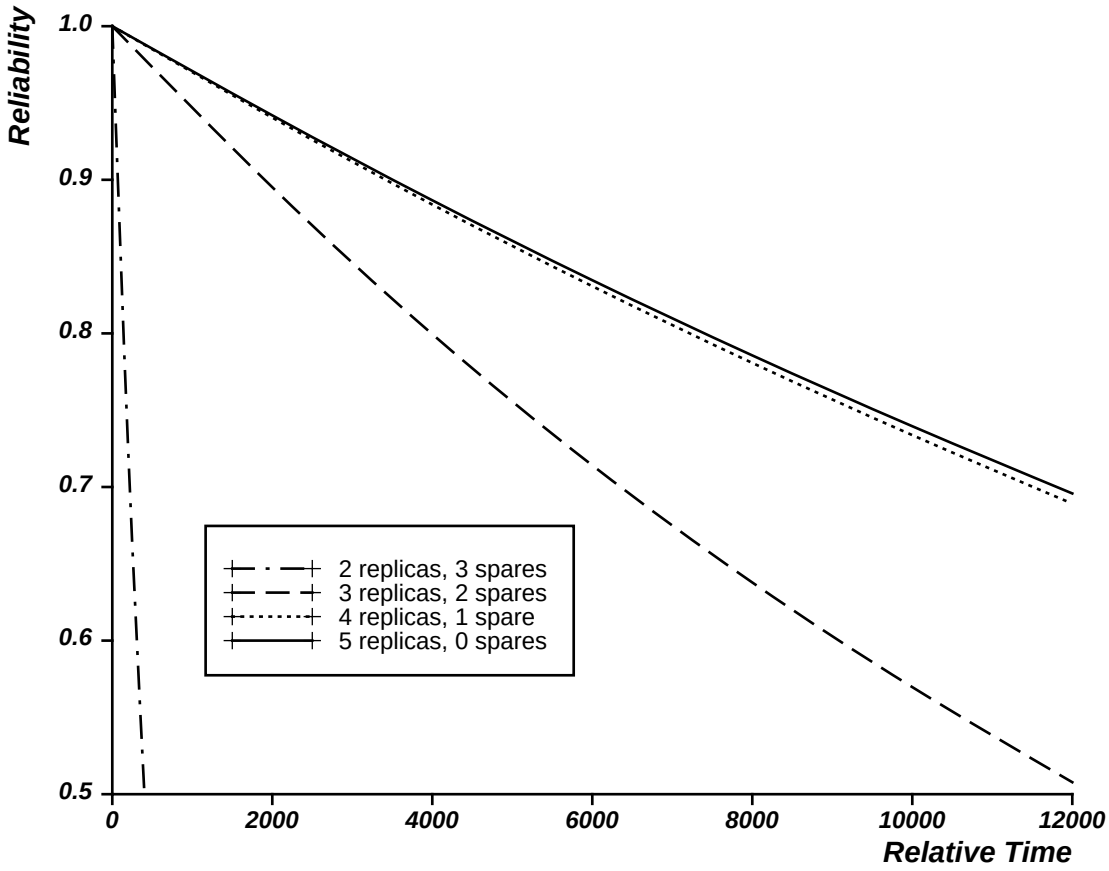

Figure 7: Compared Reliability for AC with Varying Numbers of Spares: $\kappa = 10.0, \lambda = 0.1, \mu = 1.0$

protocols, which adds more states to the analytic model. However, steady-state analysis has the advantage that the equations describing the relationships between the states of the replicated data object are not time dependent, resulting in a relatively simple set of linear equations.

Definition 5.1 *The* availability *of a replicated data object consisting of n replicas and managed by a replica control protocol P, denoted* $\mathcal{A}_P(n)$*, is the stationary probability of the system being in a state where the replica control protocol will grant access to the data object.*

The state transition diagram for a replicated data object with copies on n identical sites managed by the Available Copy protocol has $2n$ states. As in the reliability model, the n states labeled from S_1 to S_n represent the states of the data object when 1 to n copies are available. The availability model also has n states labeled from S'_0 to S'_{n-1} representing the states of the data object when all copies of the data object have failed and 0 to $n-1$ copies (not including the copy that failed) last have recovered but must remain inaccessible. As seen in Figure 8, transitions originating from

non-primed states are similar to those observed for reliability. The situation becomes different once *all* copies of the data object have failed. The data object is then in state S'_0 and will return to state S_1 if and only if the last available copy recovers. If any of the $n-1$ other copies recovers, that copy will remain *comatose* and the data object will be in state S'_1. As a result, state S'_0 has one outward transition with rate μ leading to state S_1 and another one with rate $(n-1)\mu$ leading to state S'_1.

Figure 8: State-Transition-Rate Diagram for Available Copy

All states S'_j with $j = 1, \ldots, n-2$ have three outward transitions: one leading to state S'_{j-1} corresponding to the failure j of one of the comatose copies, another one with rate μ leading to state S_{j+1} corresponding to the recovery of the last available copy, and a third one with rate $(n-j-1)\mu$ leading to state S'_{j+1} corresponding to the recovery of one of the other $n-j-1$ failed copies. State S'_{n-1} has no third outward transition since the only failed copy is necessarily the last available copy.

The resulting systems of equations for Available Copy and other protocols can be solved for small values of n, but the solutions only apply to homogeneous networks of sites characterized by exponential distributions. Simulation must be used to obtain results for situations where analytic models cannot be applied, including the investigation of how a replicated data object will behave in a real system composed of several network segments, as opposed to the simplified model used

for the Markov analysis.

5.1 Availability Model

A replicated data object is modeled as a set of replicas residing on sites distributed around the computer network. All messages are assumed to be delivered to their destinations without alterations in the order they were sent, and sites not operating correctly are assumed to immediately stop operations. Malicious failures are expressly excluded [30].

Larger local-area networks often consist of several carrier-sense networks or token rings linked by selective repeaters or gateway hosts. Since repeaters and gateways can fail without causing a total network failure, such communication networks can be partitioned. The key difference with conventional point-to-point networks is that sites that are on the same carrier-sense network or token ring will never be separated by a partition.

As discussed in §3.1, accurately determining site characteristics is a difficult task. The data used in this section was gathered at the University of California, San Diego and at the Naval Ocean Systems Center. Due to the excellent reliability of sites, it was impossible to gather enough data points to accurately determine a failure distribution, but first moments could be estimated. An exponential distribution with the experimentally determined mean is used as it accurately models the pattern of failure of most computer systems. The repair distribution of the sites is modeled as the sum of a uniformly distributed term and an exponential term.

A site is subject to several types of failure, the most common types being the failure of a system component or the failure of the operating system. A component failure will require a service call to be made and a service technician to be dispatched. Once the service technician has arrived, the problem must be diagnosed and the defective component replaced.

Sites are also subject to operating system failure. When a site suffers from such a failure, the only action required is to restart the site. Many software failures will allow the system to restart automatically in a short period of time. For the less common but more serious software failures, the dominant factor in the down-time is the amount of time it takes the system operator to notice and restart the site. This combination of many short failures and occasional longer failures is

appropriately modeled with an exponential distribution.

The sites studied are labeled A through I. The individual site characteristics are summarized in Table 1. Based on the experience of system administrators, the fraction of failures attributed to hardware is fixed at 10 percent. Sites A, E and G are VAX-11/750s, site B is a VAX 8600, sites C and D are SUN-3s, site F is a Micro-VAX, site H is a Convex and site I is a VAX-11/780. The mean-time-to-failure (MTTF) and the system restart time are based on statistics gathered at the Naval Ocean Systems Center. The percentage of failures that are attributed to defective components and the mean-time-to-repair (MTTR) required for these component failures are based on discussions with system administrators.

Table 1: Site Characteristics

Site	MTTF (hours)	MTTR U	MTTR E	Restart (minutes)
A	80.47	24.0	4.0	330.0
B	149.50	24.0	4.0	255.0
C	90.11	44.0	4.0	570.0
D	92.50	44.0	4.0	90.0
E	610.10	24.0	4.0	366.0
F	210.07	24.0	4.0	323.4
G	260.39	24.0	4.0	510.6
H	209.56	44.0	4.0	354.0
I	252.60	24.0	4.0	990.6

The communication network can be subject to different types of failure depending on the technology used. In the case of simple point-to-point networks, the links can fail. For carrier-sense segments and token rings, the only possible communication failure mode is a complete failure of the network segment. Pairs of unidirectional links connecting the various sites were chosen to model the communication network since they provide the most generality. When a site fails, all of the out-going links also fail. Although the links were not allowed to fail independently, this could also be easily modeled.

The system configuration is shown in Figure 9. Sites A, B, C, D and E are all connected to the main network segment. Site D acts as a gateway to the segment containing sites F and G. Similarly, site E acts as a gateway to the segment containing sites H and I. The result is that sites

D and E represent partition points in the system. If either of these sites fail, then the secondary segment that it serves will be isolated from the rest of the system.

Figure 9: System Configuration

The links are modeled as a binary connection matrix. An entry $s_{i,j} = 1$ appears in the connection matrix if site i can reach site j in one step. The transitive closure of this matrix represents the set of sites that can be reached in any number of steps. A *can-talk-to* relation is used to determine which sites can communicate and is implemented as

$$i \,\text{can} - \text{talk} - \text{to}\, j \Leftrightarrow s^*_{i,j} = 1 \wedge alive(j),$$

where s^* denotes the transitive closure of the connection matrix. It is necessary to ascertain if the destination site is alive since the corresponding row in the connection matrix is set to $\vec{0}$ when a site fails. This has the effect of breaking all out-going links from that site, but does not affect in-coming links. The obvious solution of setting the corresponding column in the connection matrix to $\vec{0}$ is

inappropriate since it depends on the status of all other sites. This would make the matrix very difficult to restore once a site recovers from a failure.

The simulation model is programmed in SIMSCRIPT II.5 and executed on SUN-3/60 computers. The process interaction approach [9] is used, since sites and users are easily described as independent processes.

Access to the replicated data object is modeled as a single user that can access any of the nine sites. The access requests are granted or refused based solely on the current state of the sites containing replicas of the data object and the capability of the replica control protocol to guarantee mutual consistency. Batch-means analysis [15] allows 90% confidence intervals to be computed for all performance indices. The simulations were executed for 50,000 simulated days. All sites were operating at the start of the simulation, but performance measurements only commenced after 1,000 days of operation to help ensure that only steady-state behavior is monitored.

5.2 Structure of the Availability Simulator

The availability simulation is significantly more complex than the reliability model. The reason for this is that instead of simply waiting for a site to fail, the entire life-cycle of the site must be modeled, in particular the recovery. The simulator is structured as a set of cooperating processes. A set of site processes models the behavior of the sites. The access protocols are incorporated into the single client process which models the pattern of user accesses.

5.2.1 Site Process

As in the reliability simulator, the availability simulator maintains one site process per site to model the behavior of each physical site. The site process enters a failed state after an exponentially distributed period of time. When it fails, the corresponding row of the connection matrix is set to $\vec{0}$. The transitive closure of this matrix is then computed, resulting in the removal of communication paths through the failed site. Consistency protocols with perfect (instantaneous) system configuration information are implemented by adjusting that information when a site fails.

When a failure occurs, the site process then determines the type of failure depending on a per site ratio. Failures fall into two classes: component failures, and failures of the operating system.

In the case of an operating system failure, restart times are assumed to be constant. If a service call would be required, then the process will wait a uniformly distributed period of time plus an exponentially distributed period. This is done in an effort to model the response of a service technician.

Once the site has been repaired, the corresponding row in the connection matrix is restored from the initial configuration and the transitive closure recomputed. This has the effect of restoring all paths through this site. An access check is then made to determine if the repair of this site has made the replicated data object available again. This is done for statistical purposes in order to determine the global availability of the data object. Similar requests are generated when a site fails.

5.2.2 Client Process

The client process serves several purposes. Though its primary function is to act as an accessing agent, it is also responsible for maintaining the system configuration information required by the replica control protocols. This also allows it to monitor per-site availability. Maintenance of global system availability information is distributed throughout the simulator since there are many routines in which the state of the replicated data object may change.

The time between accesses is assumed to be exponentially distributed, although any other distribution could be used if more specific distribution information is available. The client process waits for a random period of time and then chooses whether to do a read or write commensurate with the read to write ratio. The difference between read and write can be important for some protocols. A read-to-write ratio was conservatively estimated to be 4 to 1 [23]. Accesses are assumed to occur at a mean rate of once per day. In the case of Optimistic Available Copy, system configuration information is only broadcast when a write occurs. This is done to allow inexpensive read operations. In the case of Optimistic Dynamic Voting, system configuration information is maintained when any access occurs since quorum collection is always required. In keeping with the philosophy of Optimistic Available Copy and Optimistic Dynamic Voting, regenerations will occur only when a write request is made. In real systems, write requests should occur frequently enough

for regeneration to perform well.

The client process calls an access routine appropriate to the replica control protocol being modeled to determine if an access is possible from each site. One of the sites is chosen to actually perform the access operation. This also serves to maintain the system configuration information. If any of the sites can access the replicated data object, then the system is considered to be in an available state. For each of the sites that attempted an access, the per-site availability is tallied.

5.2.3 Access Routines

The access routines are implementations of the protocols as described in §2. Both the client processes and the site processes cooperate in the implementation of various parts the replica control protocols. The access procedures for each of the replica control protocols are considered next.

The access routine for Optimistic Dynamic Voting first determines the set of sites that can communicate with the site that is requesting the access. This is accomplished by consulting the can-talk-to relation for sites that hold a replica of the data object. The maximum version number and operation number are also determined at this time.

The access request will be granted if the set of sites with current operation numbers constitutes a majority of the sites in the partition set of a current replica, or represents exactly one half and have the maximal element in the total ordering on sites present in the quorum set. This is a direct implementation of the Optimistic Dynamic Voting access protocol.

Since majority consensus is a degenerate form of Optimistic Dynamic Voting that never adjusts the partition sets, this same routine is also used to determine whether accesses will be granted for that protocol. Majority consensus differs from Optimistic Dynamic Voting in that a quorum of the original number of replicas must be present in order for an access request to be granted.

The access routine for Optimistic Available Copy is simpler than the one for Optimistic Dynamic Voting. Since partitions of the communications network are impossible, it is sufficient to find any replica in an available state. In order to maintain the was-available sets required by the protocol, the access routine returns the set of available sites when a write request is granted.

Available Copy and Dynamic-linear Voting are versions of Optimistic Available Copy and Op-

timistic Dynamic Voting with perfect system configuration information. Instantaneously detecting a site failure makes the Optimistic Dynamic Voting and Available Copy protocols equivalent to the corresponding instantaneous version. An advantage of simulation is that when a site fails, the simulator can instantly react to it. Thus, a site failure will cause the partition sets, or in the case of Available Copy, the was-available sets to be adjusted accordingly. While instantaneous protocols are not implementable, they provide a good metric by which to judge the more realistic protocols.

The recovery protocols are the most complex part of the simulation. In the case of Optimistic Available Copy, the recovering site must be able to find the closure of the was-available sets, or find a replica in an available state to be able to recover. Once a site recovers to an available state, all other sites that are comatose can also be made available. The name of the new available site is then added to the was-available set of each available site in the system.

In the case of Optimistic Dynamic Voting, the recovering site must be able to communicate with the majority partition. If this is possible, then it can be made available. The name of the recovering site is added to the partition set of each site in the majority partition. The operation number is then incremented to disenfranchise sites that did not participate in the recovery.

To include regeneration in the simulation, spare sites must also be modeled. These spare sites are candidates for replicas when sites holding replicas fail. When an access occurs, it is determined whether there is a full complement of replicas available. If there is not, then a regeneration will be attempted. Replicas are placed on as many spare sites as possible, up to the original complement of replicas. The simulation could be enhanced by providing an algorithm for determining the best spare sites for regeneration.

When a site recovers from a failure, it may be that there is already a full complement of replicas. To simplify the simulation, the site is allowed to recover fully. When it is determined that there are too many replicas, some of them are returned to the pool of spares. An enhancement to the simulation would be an algorithm for determining the set of excess replicas that should be transformed into spares. As the simulator is implemented, replicas are transformed into spares according to their ordinal number. By choosing the ordering of the sites appropriately, some control over which sites will tend to remain spares can be obtained.

5.3 Results of the Study

The values obtained using various failure and repair distributions compare favorably with the results obtained using the most realistic data available. The results, even when mixing sites with disparate characteristics, closely match those obtained using an exponential distribution with a composite mean.

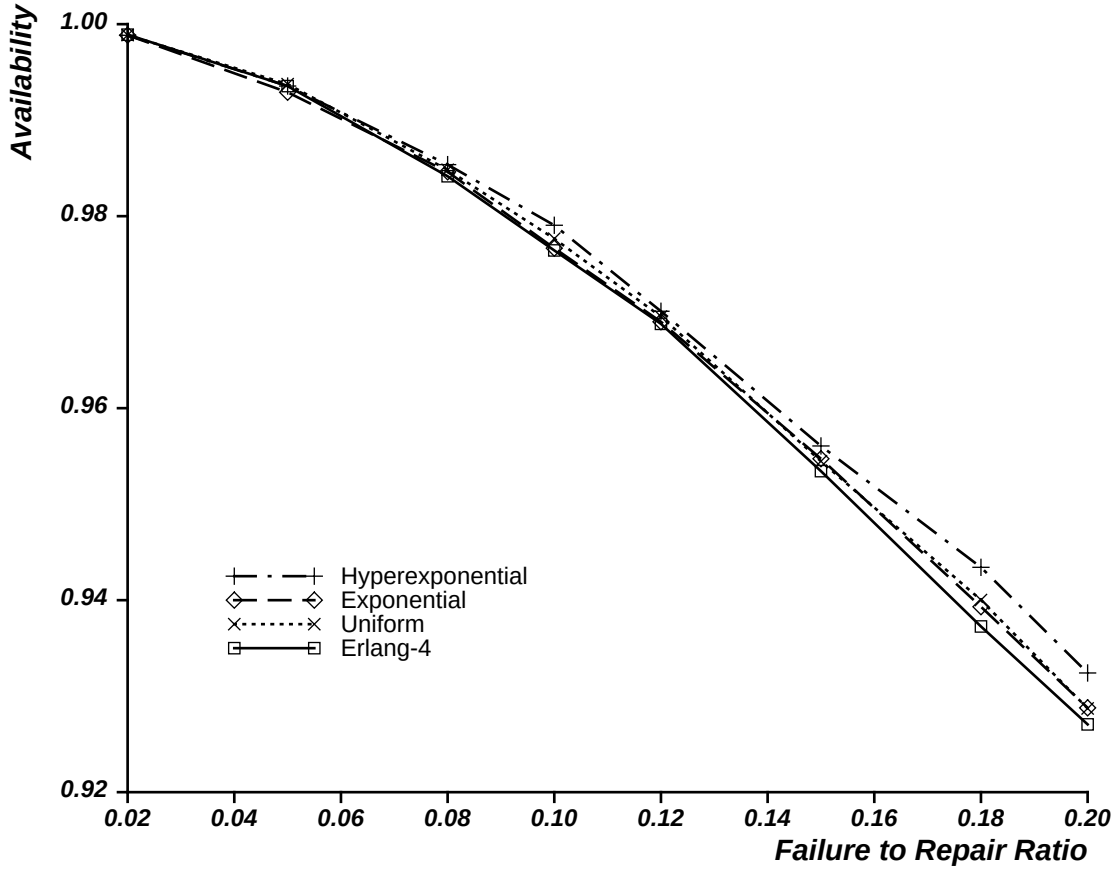

Figure 10: Availability of three sites managed by Dynamic-linear Voting

As in the reliability simulation, four representative distributions are investigated. Exponential distributions are compared with Erlang-4, uniform, and hyperexponential distributions. The "moment ratio" is again varied between 0.5 and 1.5.

Figures 10 and 11 display the availability provided by Dynamic-linear Voting and the Available Copy protocols, respectively. For both protocols, three replicas of the data object are assumed. The results are similar for larger numbers of replicas. In the case of both protocols, the effect of varying the higher moments of the failure and repair distributions is small.

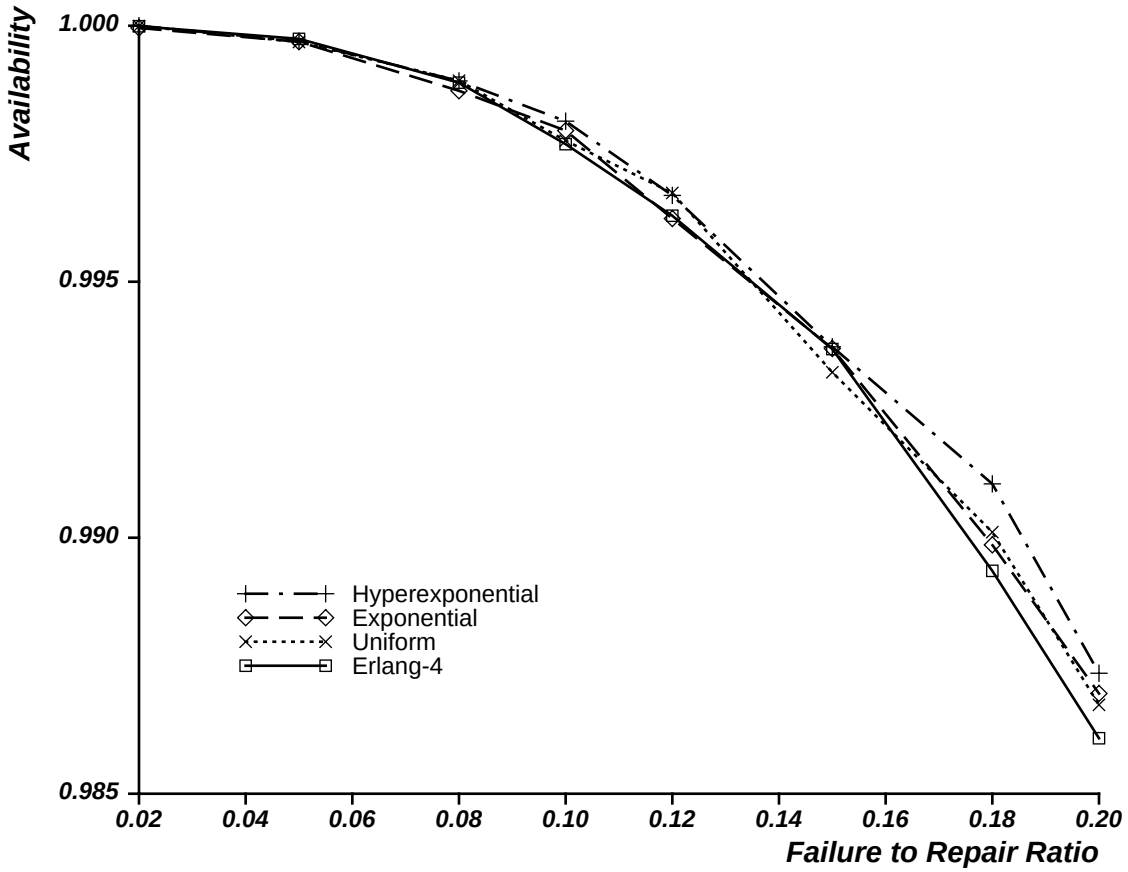

Figure 11: Availability of three sites managed by Available Copy

As the failure to repair ratio increases, the differences become more pronounced. Although not apparent in the graphs, the differences for smaller values of the failure to repair ratio are distinguishable although, as is obvious from the scale, all the availabilities are extremely high. As with reliability, an increase in the second moment leads to a slight increase in availability.

The results of the availability simulations are summarized in Tables 2 through 11. The columns of the tables indicate the configuration, the *availability,* denoted $\mathcal{A}$, that was observed for that configuration, the *unavailability,* denoted $\mathcal{U}$, which should be close to $1-\mathcal{A}$. The observed mean-time-to-failure (MTTF) of the replicated data object and the observed mean-time-to-repair (MTTR) are also measured. The unavailability is displayed since it allows the differences in the availability provided by the various protocols to be seen more clearly.

The results confirm the predictions made by other studies using Markov analysis [18], and there are consequently few surprises. The Available Copy protocol provides the highest availability

of any of the replica control protocols. As expected, the Optimistic Available Copy protocol provides availability just below that of Available Copy protocol. A similar relationship holds among instantaneous Dynamic-linear Voting, Optimistic Dynamic Voting and static Majority Consensus Voting. Dynamic-linear Voting is found to be superior, followed closely by Optimistic Dynamic Voting, while static Majority Consensus Voting trailed a distant third.

The mean-time-to-failure is highest for the Available Copy protocol, followed closely by the Optimistic Available Copy protocol. This result is not unexpected considering the reliability results obtained in §4. What is surprising is that all of the consensus protocols have very poor mean-times-to-failure. Although the reliability results indicate that they should be worse than the Available Copy protocols, the degree of difference is impressive. This can be partially explained by the way that consensus protocols recover from a total system failure. While Available Copy protocols must wait for the last copy to fail to recover, consensus protocols instead need only wait for a quorum to be present. The result is that while the consensus protocols recover more quickly, they also fail more often.

For a homogeneous network with an odd number of sites, theoretical analysis shows that adding one more site does nothing to enhance the ability to form a quorum, and hence the availability of a four-site system is identical to that of a three-site system. However, disparate site characteristics have a great deal of influence on the availability afforded by the various protocols. This can be observed clearly in the case of Majority Consensus Voting where $\mathcal{A}_{MCV}(ABC) < \mathcal{A}_{MCV}(ABCD)$, which would seem to contradict predictions made by Markov analysis. But, if the site characteristics are considered, it can be seen that site D is more accessible than site C. This illustrates the effect of different site characteristics on the performance of the replica control protocol, since site D overshadows the poor behavior of site C.

The regeneration results are also of interest. As defined in the protocols, regenerations occur only when a write request is made. Since accesses occurred at the conservative rate of once per day, and with the read to write ration of 4 to 1, regeneration is found to have only a small effect on the performance of the protocols. A useful extension to this work would be to determine a more accurate estimates for the access frequency and study how this affects the regeneration process.

In some cases, regeneration can have a negative influence on the performance of the protocols. This is most easily seen in the case of Optimistic Available Copy. Tables 4 and 5 show that the availability of two sites $\mathcal{A}_{OAC}(AB)$ is greater than $\mathcal{A}_{OAC}(AB, C)$, which also employs C as a spare. The explanation for this again involves the characteristics of site C and the way in which replicas are transformed into spares. The simulator transforms replicas into spares according to their lexicographic order, so site A will be transformed into a spare and the resulting configuration performs much like BC. In general, regeneration does have a positive effect on the performance of the replica control protocols, and considering its low cost due to the efficient implementations of Optimistic Available Copy and Optimistic Dynamic Voting, it should be considered as a viable alternative to keeping extra replicas of a data object.

Although only applicable to the consensus protocols, partitions of the communication network are found to have a significant effect on those replica control protocols. Consider, for example, the Dynamic-linear Voting protocol. $\mathcal{A}_{DLV}(ABCD) > \mathcal{A}_{DLV}(ABCDFG)$ since sites F and G are on the other side of site D with respect to sites A, B and C. When site D fails, sites F and G are excluded from most quorums by the network partition. The exception to this is when sites F and G could form a majority partition, but this is a rare event.

There are some limitations of the simulation model. It was not feasible to execute the simulations long enough to get accurate measures for some statistics. In particular, Available Copy with five replicas did not fail in over 100,000 simulated days. Similarly, it was not possible to model the regeneration protocols for a wide range of access rates due to the excessive computation required.

Table 2: Available Copy

Configuration Replicas	$\mathcal{A}$	$\mathcal{U}$	MTTF (Days)	MTTR (Days)
AB	0.994200	0.005781	45.21003	0.26394
ABC	0.999097	0.000907	314.24409	0.28374
ABCD	0.999937	0.000063	4397.56470	0.28486
CD	0.990716	0.009300	26.39872	0.24778
CDE	0.999756	0.000245	1369.40235	0.33168
FG	0.997800	0.002204	146.38771	0.32152

Table 3: Available Copy with Regeneration

Configuration		$\mathcal{A}$	$\mathcal{U}$	MTTF	MTTR
Replicas	Spares			(Days)	(Days)
AB	C	0.99220	0.00776	40.55821	0.31830
AB	CD	0.99124	0.00877	27.75885	0.24520
AB	CDE	0.99600	0.00401	61.33133	0.24668
ABC	D	0.99967	0.00033	552.60274	0.18353
ABC	DE	0.99976	0.00024	1521.23026	0.35932
CD	A	0.99120	0.00880	27.71331	0.24588
CD	AB	0.99124	0.00877	27.75885	0.24520
CDE	A	0.99976	0.00024	1521.23026	0.35932
CDE	AB	0.99976	0.00024	1521.23026	0.35932

Table 4: Optimistic Available Copy

Configuration	$\mathcal{A}$	$\mathcal{U}$	MTTF	MTTR
Replicas			(Days)	(Days)
AB	0.992539	0.007450	46.18184	0.34749
ABC	0.998442	0.001564	318.10792	0.49561
ABCD	0.999844	0.000157	4397.10728	0.60998
CD	0.982757	0.017276	27.77548	0.48783
CDE	0.999578	0.000424	1369.16256	0.57522
FG	0.997475	0.002531	146.57526	0.37065

6 Conclusions

Two simulation models aimed at comparing the performance of several consistency control protocols for replicated data objects in real-life situations have been presented. By using discrete event simulation, the results obtained using Markov models can be validated and the robustness of the exponential approximations judged. The distributions investigated here cover a wide range of second moments, and clearly support the contention that the actual system distributions will produce results that are consistent with the behavior predicted by exponential distributions. This enhances the importance and applicability of the results obtained by theoretical analysis, which depend on Markovian assumptions.

Since no replicated object can ever be accessed without having at least one current replica, Available Copy protocols are known to provide the highest possible reliability figures of all consistency protocols that do not incorporate new sites to replace the ones that have failed [27]. Available

Table 5: Optimistic Available Copy with Regeneration

Configuration		$\mathcal{A}$	$\mathcal{U}$	MTTF	MTTR
Replicas	Spares			(Days)	(Days)
AB	C	0.98952	0.01045	41.36196	0.43969
AB	CD	0.98327	0.01675	29.26129	0.49798
AB	CDE	0.99243	0.00757	64.36243	0.49114
ABC	D	0.99899	0.00100	567.66332	0.58346
ABC	DE	0.99959	0.00040	1520.97802	0.61577
CD	A	0.98323	0.01679	29.13638	0.49729
CD	AB	0.98327	0.01675	29.26129	0.49798
CDE	A	0.99959	0.00040	1520.97802	0.61577
CDE	AB	0.99959	0.00040	1520.97802	0.61577

Table 6: Majority Consensus

Configuration	$\mathcal{A}$	$\mathcal{U}$	MTTF	MTTR
Replicas			(Days)	(Days)
ABC	0.984890	0.015109	10.661674	0.16361
ABCD	0.990676	0.009311	12.396540	0.11668
ABCDE	0.998766	0.001238	74.102941	0.09166
ABCFG	0.989508	0.010492	10.983852	0.11668

Copy also has the highest availability of any of these protocols. While the availabilities of the various schemes are somewhat similar, the reliabilities differ by orders of magnitude. Where reliability is important, the Available Copy protocols are clearly superior [21]. It is clear from Figures 3, 4 and 5 that the shapes of the failure and repair distributions in no way affect the relative merits of the protocols.

However, it must be noted that the Available Copy protocol cannot guarantee consistency in the presence of network partitions, and is therefore inappropriate in environments in which partial communication failures are possible, such as when sites are separated by gateways. The Available Copy protocol is guaranteed to function correctly if all replicas of the data object are stored on the same carrier sense segment or token ring, and hence it is both appropriate and desirable in many applications. Voting protocols guard against the partitioning problem at the expense of performance.

In the absence of contrived circumstances, the first moments of the failure and repair distributions are the overwhelmingly predominant factors. With equal first moments, larger second

Table 7: Majority Consensus with Regeneration

Configuration		$\mathcal{A}$	$\mathcal{U}$	MTTF	MTTR
Replicas	Spares			(Days)	(Days)
ABC	D	0.989190	0.010785	13.595055	0.14860
ABC	DE	0.993083	0.006938	21.542634	0.15029
ABC	F	0.978791	0.021194	9.064501	0.19637
ABC	FH	0.973494	0.026467	8.239005	0.22462
ABCD	E	0.997603	0.002406	68.810735	0.165538
ABCD	F	0.954368	0.045641	3.528837	0.168738
ABCD	FG	0.991841	0.008182	22.171667	0.182710
ABCD	FH	0.983256	0.016719	10.811329	0.184184
ABCDE	F	0.992561	0.007448	17.675419	0.13255
ABCDE	FG	0.958569	0.041435	3.859547	0.16682
ABCDE	FH	0.989841	0.010163	15.067668	0.15350
ABCDE	H	0.997035	0.002978	52.378700	0.15609

Table 8: Dynamic-linear Voting

Configuration	$\mathcal{A}$	$\mathcal{U}$	MTTF	MTTR
Replicas			(Days)	(Days)
ABC	0.983699	0.016284	23.33426	0.38660
ABCD	0.998548	0.001459	168.50073	0.24714
ABCDE	0.999950	0.000051	9958.99994	0.51857
ABCFG	0.998496	0.001498	251.46865	0.37801

moments slightly increase performance, and third and higher moments seem to have no measurable effect.

It is possible to postulate very unrealistic distributions that yield results at variance with the exponential predictions. Sites with identical constant failure and repair distributions in which the failures were synchronized could result in accessibility that was no better than that provided with a single site. On the other hand, constant distributions in which the failures were staggered could instead ensure perfect reliability and availability. In a similar fashion, using distributions with very small variances can result in reliability that is either insignificantly better than a single site or almost perfectly reliable, depending on the initial staggering of the distributions.

Discrete event simulation also provided the ability to study more realistic scenarios using data obtained from real systems. While each of a wide range of distributions behaved similarly to the exponential distributions, the realistic data yielded results that were gratifyingly close to the

Table 9: Dynamic-linear Voting with Regeneration

Configuration		$\mathcal{A}$	$\mathcal{U}$	MTTF	MTTR
Replicas	Spares			(Days)	(Days)
ABC	D	0.989702	0.010312	22.83561	0.23761
ABC	DE	0.997950	0.002057	127.64961	0.26223
ABC	F	0.974753	0.025315	16.35642	0.42401
ABC	FH	0.987619	0.012289	25.91903	0.32456
ABCD	E	0.999701	0.000301	1014.09271	0.30029
ABCD	F	0.959734	0.040268	3.99966	0.16781
ABCD	FG	0.996391	0.003619	55.69994	0.20191
ABCD	FH	0.991328	0.008684	21.14195	0.18590
ABCDE	F	0.993604	0.006394	28.48776	0.18326
ABCDE	FG	0.963879	0.036127	4.46090	0.16717
ABCDE	FH	0.994053	0.005963	37.46007	0.22449
ABCDE	H	0.997939	0.002070	146.55367	0.30345

Table 10: Optimistic Dynamic Voting

Configuration	$\mathcal{A}$	$\mathcal{U}$	MTTF	MTTR
Replicas			(Days)	(Days)
ABC	0.985639	0.014356	12.478452	0.18180
ABCD	0.994026	0.005982	17.300709	0.10397
ABCDE	0.999246	0.000755	117.288427	0.08840
ABCFG	0.992892	0.007110	15.486834	0.11073

exponential approximations.

The analysis of the trade-off between storage space and reliability showed that replacing a single replica managed by Available Copy in a five-machine system with a spare has only a slight detrimental effect on the accessibility of the data object, but further decreasing the number of replicas markedly degrades the reliability. Similar limits are reached for the other protocols as well.

The simulation results, backed by numerical solutions of the Markov models, 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

Table 11: Optimistic Dynamic Voting with Regeneration

Configuration		$\mathcal{A}$	$\mathcal{U}$	MTTF	MTTR
Replicas	Spares			(Days)	(Days)
ABC	D	0.990861	0.009144	14.961130	0.137975
ABC	DE	0.995107	0.004906	26.259576	0.129345
ABC	F	0.98295	0.017116	10.925013	0.190318
ABC	FH	0.98558	0.014384	11.666053	0.170915
ABCD	E	0.99823	0.001774	81.958399	0.145592
ABCD	F	0.95599	0.044024	3.617164	0.166539
ABCD	FG	0.99364	0.006384	25.149958	0.161096
ABCD	FH	0.98881	0.011198	12.998360	0.147150
ABCDE	F	0.99453	0.005468	20.584974	0.113211
ABCDE	FG	0.95981	0.040191	3.973005	0.166390
ABCDE	FH	0.99259	0.007415	18.485102	0.138063
ABCDE	H	0.99761	0.002391	63.271525	0.151308

level of reliability.

Regeneration was found to improve the performance of all replica control protocols for most configurations. Due to computational constraints, insufficient simulation data was available to adequately determine the effect of the access rate on the regeneration process. This should provide an interesting area for future research.

Acknowledgements

The authors are grateful to Jehan-François Pâris, Walter Burkhard, Kris Stewart, Alexander Glockner, Mary Long, Ernestine M^{c}Kinney, Susan Hudson and Robin Fishbaugh 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] D. Barbara, H. Garcia-Molina, and A. Spauster, "Policies for dynamic vote reassignment," in *Proceedings* 6^{th} *International Conference on Distributed Computing Systems*, (Cambridge), pp. 37–44, IEEE, 1986.

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

[3] P. A. Bernstein, V. Hadzilacos, and N. Goodman, *Concurrency Control and Recovery in Database Systems*. Reading, Massachusetts: Addison-Wesley, 1987.

[4] P. A. Bernstein and D. W. Shipman, "The correctness of concurrency control mechanisms in a systems for distributed databases — (SDD-1)," *ACM Transactions on Database Systems*, vol. 5, March 1980.

[5] P. A. Bernstein, D. W. Shipman, and J. B. Rothnie, "Concurrency control in a system for distributed databases (SDD-1)," *ACM Transactions on Database Systems*, vol. 5, pp. 18–51, March 1980.

[6] J. L. Carroll, D. D. E. Long, and J.-F. Pâris, "Block-level consistency of replicated files," in *Proceedings of the* 7^{th} *International Conference on Distributed Computing Systems*, (Berlin), pp. 146–153, IEEE, September 1987.

[7] D. Davčev and W. A. Burkhard, "Consistency and recovery control for replicated files," in *Proceedings of the* 10^{th} *Symposium on Operating Systems Principles*, (Orcas Island), pp. 87–96, ACM, 1985.

[8] C. A. Ellis, "Consistency and correctness of duplicate database systems," *Operating Systems Review*, vol. 11, 1977.

[9] G. S. Fishman, *Principles of Discrete Event Simulation*. New York: Wiley and Sons, 1978.

[10] H. Garcia-Molina and D. Barbara, "How to assign votes in a distributed system," *Journal of the Association for Computing Machinery*, vol. 32, pp. 841–855, October 1985.

[11] D. K. Gifford, "Weighted voting for replicated data," in *Proceedings* 7^{th} *Symposium on Operating System Principles*, (Pacific Grove), pp. 150–161, ACM, 1979.

[12] S. Jajodia, "Managing replicated files in partitioned distributed database systems," in *Proceedings* 3^{rd} *International Conference on Data Engineering*, (Los Angeles), pp. 412–418, IEEE, 1987.

[13] S. Jajodia and D. Mutchler, "Dynamic voting," in *SIGMOD International Conference on Data Management*, pp. 227–238, ACM, 1987.

[14] S. Jajodia and D. Mutchler, "Integrating static and dynamic voting protocols to enhance file availability," in *Proceedings* 4^{th} *International Conference on Data Engineering*, (Los Angeles), pp. 144–153, IEEE, 1988.

[15] A. Law and R. Mills, *Statistical Analysis of Simulation Output Data Using SIMSCRIPT II. 5*. Los Angeles: CACI, 1988.

[16] D. D. E. Long, *The Management of Replication in a Distributed System*. Ph.D. dissertation, University of California, San Diego, August 1988.

[17] D. D. E. Long, "On the storage requirements of regeneration," Tech. Rep. UCSC-CRL–89–04, University of California, Santa Cruz, July 1989.

[18] D. D. E. Long, J. L. Carroll, and K. Stewart, "Estimating the reliability of regeneration-based replica control protocols," *IEEE Transactions on Computers*, December 1989.

[19] D. D. E. Long and J.-F. Pâris, "On improving the availablility of replicated files," in *Proceedings of the* 6^{th} *Symposium on Reliable Distributed Systems*, (Williamsburg), pp. 77–83, IEEE, March 1987.

[20] D. D. E. Long and J.-F. Pâris, "A realistic evaluation of Optimistic Dynamic Voting," in *Proceedings of the* 7^{th} *Symposium on Reliable Distributed Systems*, (Columbus), pp. 129–137, IEEE, October 1988.

[21] D. D. E. Long, J.-F. Pâris, and J. L. Carroll, "Reliability of replicated data objects," in *Proceedings of the* 8^{th} *International Conference on Computers and Communications*, (Phoenix), pp. 402–406, IEEE, March 1989.

[22] J. D. Noe and A. Andreassian, "Effectiveness of replication in distributed computing networks," in *Proceedings of the* 7^{th} *International Conference on Distributed Computing Systems*, (Berlin), pp. 508–513, IEEE, 1987.

[23] J. Ousterhout, H. Da Costa, D. Harrison, J. Kunze, M. Kupfer, and J. Thompson, "A trace-driven analysis of the UNIX 4. 2 BSD file system," in *Proceedings* 10^{th} *Symposium on Operating System Principles*, (Orcas Island), pp. 15–24, ACM, 1985.

[24] J.-F. Pâris, "Voting with a variable number of copies," in *Proceedings* 16^{th} *Fault-Tolerant Computing Symposium*, (Vienna), pp. 50–55, IEEE, 1986.

[25] J.-F. Pâris, "Voting with witnesses: A consistency scheme for replicated files," in *Proceedings of the* 6^{th} *International Conference on Distributed Computing Systems*, (Cambridge), pp. 606–612, IEEE, 1986.

[26] J.-F. Pâris, "Efficient management of replicated data," in *Proceedings of the International Conference on Database Theory*, (Bruges, Belgium), 1988.

[27] J.-F. Pâris and D. D. E. Long, "The performance of available copy protocols for the management of replicated data," *Performance Evaluation*, 1990. to appear.

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

[29] C. Pu, J. D. Noe, and A. Proudfoot, "Regeneration of replicated objects: A technique and its Eden implementation," in *Proceedings of the* 2^{nd} *International Conference on Data Engineering*, (Los Angeles), pp. 175–187, IEEE, 1986.

[30] R. D. Schlichting and F. B. Schneider, "Fail stop processors: An approach to designing fault-tolerant computing systems," *ACM Transactions on Computer Systems*, vol. 1, pp. 222–238, 1983.

[31] D. Skeen, "A quorum-based commit protocol," in *Proceedings of the* 6^{th} *Berkeley Workshop on Distributed Data Management and Computer Networks*, (Berkeley), pp. 69–80, University of California, 1982.

[32] D. Skeen, "Determining the last process to fail," *ACM Transaction on Computer Systems*, vol. 3, pp. 15–30, 1985.

[33] R. H. Thomas, "A majority consensus approach to concurrency control," *ACM Transactions on Database Systems*, vol. 4, pp. 180–209, 1979.

[34] K. S. Trivedi, *Probability & Statistics with Reliability, Queuing and Computer Science Applications*. Englewood Cliffs, New Jersey: Prentice-Hall, 1982.

Appendix

```
''
'' Reliability tests: with regeneration, finite spares.  Uses all exponential
'' distributions for validation against numeric solutions.  Other distributions
'' have also been used.
''
'' Detection of down site and replacement of the dead site averages 1/kappa
'' time units.  1/mu is the average total time to both repair and regenerate.
'' Independent random number streams are used for each type of state change.
''
'' The program records length of time until each protocol would declare files
'' to be unavailable due to site failures.  Partitions are not modelled.
''
'' Output consists of failure times which are then sorted to obtain deciles.
''
'' Protocols investigated:
''
''   Available Copy
''   Dynamic Voting
''   Dynamic-linear Voting
''   Majority Consensus Voting
''
'' (c) JLC & DDEL, 1988.
''

preamble
    normally, mode is undefined

events include NEW
    every REGENERATION has a CANDIDATE

''
'' Sites and spares exist in one of two states: up or down.
''
processes include SITE
    every SITE may belong to the UP.SITES
    every SITE may belong to the UP.SPARES
    every SITE may belong to the DOWN.SITES
    every SITE may belong to the DOWN.SPARES

    the system owns the UP.SITES
    the system owns the UP.SPARES
    the system owns the DOWN.SITES
    the system owns the DOWN.SPARES

define I,         '' A trivial loop counter.
```

```
       N,         '' The number of sites.
       S,         '' The number of spares.
       Nd2,       '' div.f(N,2) done once to save time.
       DONE,      '' The number of iterations of simulation completed thus far.
       ITER,      '' The number of iterations requested.
       CANDIDATE  '' A dummy variable for event regeneration.
       as integer variables

define MTTF,      '' The mean time to failure (1/lambda).
       MTTR,      '' The mean time to repair (1/mu).
       MTTR.REG,  '' The mean time to regenerate (1/kappa).
       LAST.FAIL  '' The time the last iteration failed.
       as real variables

define UP       as a 1-dim integer array  '' Is a protocol still functional?
define UP.TIME  as a 1-dim real    array  '' For how long?

define SITES.LEFT to mean N.UP.SITES

define FALSE to mean 0
define TRUE  to mean 1

''
'' Protocol numbers.
''
define AC    to mean 1  '' Available Copy
define DV    to mean 2  '' Dynamic Voting
define DLV   to mean 3  '' Dynamic-linear Voting
define MCV   to mean 4  '' Majority Consensus Voting

''
'' Files.
''
define UNIT.IN   to mean 09
define GEN.OUT   to mean 10
define STAND.OUT to mean 06

end '' PREAMBLE

main

''
'' There are 4 protocols to consider.
''
reserve UP as 4
reserve UP.TIME as 4
```

```
''
'' How many sites, excluding spares?
''
print 1 line thus
Enter the number of sites:
read N

''
'' To simulate without regeneration, specify zero spares.
''
print 1 line thus
Enter the number of spares:
read S

Nd2 = div.f(N,2)    '' Integer divide, done once to save time.

print 1 line thus
Enter the number of iterations:
read ITER

''
'' Read in the system parameters.
''
use UNIT.IN for input
read MTTF, MTTR, MTTR.REG

print 5 lines with N, S, Nd2, MTTF, 1/MTTF,
MTTR, 1/MTTR, MTTR.REG, 1/MTTR.REG, MTTR/MTTF thus
There are ** sites, ** spares; majority consensus fails when ** sites are up.
the mean time to fail       is  **.***  => lambda = *.***
the mean time to repair     is  **.***  => mu     = *.***
the mean time to regenerate is  **.***  => kappa  = *.***
the ratio lambda/mu = MTTR/MTTF     => rho    = *.***

use GEN.OUT for output
print 5 lines with N, S, Nd2, MTTF, 1/MTTF,
MTTR, 1/MTTR, MTTR.REG, 1/MTTR.REG, MTTR/MTTF thus
There are ** sites, ** spares; majority consensus fails when ** sites are up.
the mean time to fail       is  **.***  => lambda = *.***
the mean time to repair     is  **.***  => mu     = *.***
the mean time to regenerate is  **.***  => kappa  = *.***
the ratio lambda/mu = MTTR/MTTF     => rho    = *.***
use STAND.OUT for output

schedule a NEW now
start simulation
```

```
end '' MAIN

''
'' NEW -- Collect statistics and restore the system to its initial state.
''
event NEW

''
'' Reset the state of each protocol.
''
for I = 1 to 4 do
    UP(I)      = TRUE
    UP.TIME(I) = 0.0
loop

''
'' Are we finished yet?
''
if ITER > DONE

    LAST.FAIL = time.v

    ''
    '' New sites.
    ''
    for I = 1 to N do
        activate a SITE now
        file this SITE in UP.SITES
    loop

    ''
    '' New spares.
    ''
    for I = 1 to S do
        activate a SITE now
        file this SITE in UP.SPARES
    loop

    DONE = DONE + 1

always

end '' NEW

''
'' REGENERATION -- Transform a spare into a full site.
''
```

```
event REGENERATION given THIS.SITE
define THIS.SITE,
       ALT.SITE   '' The next candidate for regeneration.
       as integer variables

''
'' When a total failure has occured, regeneration is impossible.
''
if UP(AC) = FALSE
    return
always

''
'' If the candidate is still alive, make it a site.
''
if THIS.SITE is in UP.SPARES
    remove THIS.SITE from UP.SPARES
    file THIS.SITE in UP.SITES
else
''
'' Attempt to regenerate on another site.
''
    for each ALT.SITE of UP.SPARES
        find the first case
        ''
        '' Note: this skews regeneration towards the lower numbered sites.
        ''
        if found,
            schedule a REGENERATION giving ALT.SITE in
            exponential.f(MTTR.REG, 4) units
        always
always

end '' REGENERATION

''
'' SITE -- each time a site (or spare) fails, it is switched to DOWN.SITES
'' (DOWN.SPARES); checks are made to see if each protocol has failed, and the
'' total length of the uptime is then recorded.
''
'' Each site process exits the loop as soon as all protocols have died; the last
'' remaining process starts a new iteration.
''
process SITE

define DLV.DEAD,  '' For Dynamic-linear Voting.
       ALT.SITE   '' Candidate for regeneration.
```

```
       as integer variables

''
'' When Available Copy dies, there is a total failure.
''
until UP(AC) = FALSE do
    wait exponential.f(MTTF,1) units '' Work a while then die.

    ''
    '' If it was just a spare, don't check.
    ''
    if SITE in UP.SPARES
        remove this SITE from UP.SPARES
        file this SITE in DOWN.SPARES
    else
        remove this SITE from UP.SITES
        file this SITE in DOWN.SITES

        ''
        '' Has Dynamic Voting failed?
        ''
        if UP(DV) = TRUE
            ''
            '' We need at least 2 sites to continue.
            ''
            if SITES.LEFT < 2
                UP.TIME(DV) = time.v - LAST.FAIL
                UP(DV) = FALSE
            always
        always

        ''
        '' Has majority consensus failed?
        ''
        if UP(MCV) = TRUE
            ''
            '' More than half have died.
            ''
            if SITES.LEFT <= Nd2
                UP.TIME(MCV) = time.v - LAST.FAIL
                UP(MCV) = FALSE
            always
        always

        ''
        '' Has Dynamic-linear Voting failed?
        ''
```

```
    if UP(DLV) = TRUE
        ''
        '' Dynamic-linear Voting will survive 50% of the time when one
        '' of two remaining sites fail, due to lexicographic ordering.
        ''
        if SITES.LEFT = 1
            DLV.DEAD = randi.f(1,2,3)
            if DLV.DEAD = TRUE
                UP.TIME(DLV) = time.v - LAST.FAIL
                UP(DLV) = FALSE
            always
        ''
        '' The last site has died.
        ''
        else if SITES.LEFT = 0
            UP.TIME(DLV) = time.v - LAST.FAIL
            UP(DLV) = FALSE
        always
        always
    always

    ''
    '' If all active sites are down, then Available Copy fails, and we
    '' are done with this iteration.
    ''
    if SITES.LEFT = 0
        UP.TIME(AC) = time.v - LAST.FAIL
        UP(AC) = FALSE
    else
        ''
        '' Attempt to regenerate on another site.
        ''
        for each ALT.SITE of UP.SPARES
            find the first case
            ''
            '' Note: this skews regeneration towards the lower numbered
            '' sites.
            ''
            if found,
                schedule a REGENERATION giving ALT.SITE in
                exponential.f(MTTR.REG,4) units
            always
    always
always

''
'' Repair the site after an appropriate down time.
```

```
    ''
    wait exponential.f(MTTR,2) units

    if SITE is in DOWN.SITES
        remove this SITE from DOWN.SITES
    else
        remove this SITE from DOWN.SPARES
    always

    if N.UP.SITES < N
        file this SITE in UP.SITES
    else
        file this SITE in UP.SPARES
    always
loop

if SITE is in UP.SITES
    remove this SITE from UP.SITES
else if SITE is in UP.SPARES
    remove this SITE from UP.SPARES
always
always

''
'' Is this the last site to notice that there has been a total failure?
''
if N.DOWN.SITES + N.DOWN.SPARES + N.UP.SITES + N.UP.SPARES = 0
''
'' The last remaining site process is responsible for printing the results of
'' this iteration.
''
'' Write the results to files:
'' 1 =  AC: Available Copy
'' 2 =  DV: Dynamic Voting
'' 3 = DLV: Dynamic-linear Voting
'' 4 = MCV: Majority Consensus Voting
''
for I = 1 to 4 do
    use I for output
    print 1 line with UP.TIME(I) thus
**********.****
loop

use STAND.OUT for output

schedule a NEW now
```

```
always

end '' SITE
```