1-1-1990

Effects of Replication on Data Availability

Donald B. Johnson
Dartmouth College

Larry Raab
Dartmouth College

EFFECTS OF REPLICATION ON DATA AVAILABILITY

Donald B. Johnson and Larry Raab

Technical Report PCS-TR90-155

Effects of Replication on Data Availability

Donald B. Johnson* Larry Raab†
Dartmouth College‡

Abstract

In this paper we examine the effects of replication on the availability of data in a large network. This analysis differs from previous analyses in that it compares the performance of a dynamic consistency control protocol not only to that of other consistency control protocols, but also to the performance of non-replication and to an upper bound on data availability. This analysis also differs in that we gather extensive simulations on large networks subject to partitions at realistically high component reliabilities. We examine the dynamic consistency control protocol presented by Jajodia and Mutchler[9, 12] and by Long and Pâris[18] along with two proposed enhancements to this protocol[10, 11]. We study networks of 101 sites and up to 5050 links (fully-connected) in which all components, although highly reliable, are subject to failure. We demonstrate the importance in this realistic environment of an oft neglected parameter of the system model, the ratio of transaction submissions to component failures. We also show the impact of the number of copies on both the protocol performance and the potential of replication as measured by the upper bound. Our simulations show that the majority of current protocol performs optimally for topologies that yield availabilities of at least 65%. On the other hand, the availability provided by non-replication is inferior to that of the majority of current protocol by at most 5.9 percentage points for these same topologies. At this point of maximum difference, the primary copy protocol yields availability 59.1% and the majority of current protocol yields availability 65.0%. We discuss the characteristics of the model limiting the performance of replication.

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

1. Introduction

In a distributed system, data replication can be used to improve the availability of data objects. Traditionally, this increase in availability has been attributed to an increase in the probability that at least one copy is accessible due to the additional copies. This is considered especially helpful during a network partition, that is when the network contains disjoint components due to site or link failures. After presenting our simulation results, we will examine more fully the sources of increased availability due to replication.

The full benefits of data replication are in practice difficult to achieve due to the need for data *correctness*. If we define a *transaction* as an atomic series of data accesses, then by correctness we mean that the concurrent execution of transactions on replicated objects is equivalent to some serial execution on non-replicated data; this constraint is known as *one-copy serializability*[4]. It is the task of a concurrency control protocol to guarantee one-copy serializability in the face of component failures and consequent network partitioning. That portion of a concurrency control function that guarantees the *mutual consistency* of replicated data objects is the consistency control protocol.

Partition resilient consistency control protocols that help guarantee one-copy serializability while never requiring a transaction to be reversed do so by enforcing what we will call the consistency constraints. A *component* at a point in time is a maximal set of sites which can communicate. A *distinguished* component is a component from within which a site is allowed to update the data object. For the purposes of this paper, we assume that if any copy of a data item is updated, every copy in the component is updated simultaneously. Using these definitions, the consistency constraints can be stated as follows: (1) no two distinguished components may exist simultaneously, and (2) the set of copies in two successive distinguished components must have at least one copy in common.

Our results are unique in this area in that we have extensively simulated models that are analytically intractable. These models capture essential features of network performance, particularly the formation and coalescence of components, for which there are no suitable analytic techniques known. We consider networks of 101 sites and up to 5050 links (fully-connected). We analyze the majority of current dynamic consistency control protocol along with two proposed enhancements, linear ordering and majority consensus hybrid. Before citing relative performance figures, we demonstrate the sensitivity of these figures to the ratio of transaction submissions to component

failures and recovers. This ratio, which is infrequently mentioned and never discussed to our knowledge in the literature, proves to be critical to the understanding of the performance of any dynamic consistency control protocol. Unlike other analyses, we also compare the performance of this replication protocol to that of a single copy and to our optimal oracle protocol. In so doing, we provide insight into the potential benefits of replication as a method of increasing availability.

The system we consider is composed of sites and bi-directional links. Links fail by failing to transmit messages; partial failures such as links operating in only one direction or garbling messages are not considered. Our assumption is that any message transmitted is correct in its entirety and that the sequential order of transmitted messages is preserved. Processors are fail-stop [19]; although they may fail to send or receive a message, byzantine failures[13] are not considered. Since message passing is the only inter-node communication mechanism, processor and link failures can partition the network into disjoint sets of communicating sites called *components*. When communication between two processors is lost it is impossible for one site to determine whether the other has failed or is partitioned away. Finally, all node and link failures are eventually repaired, although once repaired they are again subject to failure.

The remainder of this paper is organized as follows. We begin with a discussion of our performance metrics, followed in section 3 by a brief description of the protocols we analyze. These include the primary copy protocol[2], the majority consensus protocol[20], and the majority of current protocol (MOC)[9, 12, 18] along with two enhancements[10, 11]. The oracle protocol used to provide an upper bound on data availability is introduced in section 3.4. Section 4 deals with the parameters and assumptions of our simulator. The significance of one of these parameters, the ratio of transaction submissions to component failures, is explained in section 5.1. In section 5.2 we compare a number of combinations of MOC enhancements, and in section 5.3 we compare MOC to the primary copy protocol and to our oracle upper bound. We conclude with a discussion of the performance of replication in general.

2. Measures of Data Availability

There are two definitions of data availability in the literature[10, 12]. The metric more commonly used, which we will call *Survivability*($SURV$), is the probability that at an arbitrary time there exists at least one site which

may access the data object. This is equivalent to the probability that a distinguished component exists or to one minus the probability of a halting state. The second metric, which we will call *Accessibility*(ACC), is the probability that at an arbitrary time an arbitrary site may access the data object.

The difference between these two metrics is that $SURV$ measures the amount of time that a data object can be accessed from some site in the network, while ACC is a measure of the number of sites which are able to perform that access. As a result, an increase in the number of sites that can access the data item will result in an increase in ACC, but $SURV$ will remain unchanged. Thus $SURV$ favors dynamic protocols since they tend to produce smaller distinguished components than do the static protocols. More importantly, the reliability of a single site is a lower bound for $SURV$, since $SURV$ is always realizable by a single copy, and an upper bound for ACC, since at least the submit site must be up. We have chosen to present our results using the ACC metric since we are interested in the probability that an arbitrary site will be able to access the data item and therefore would not expect availability greater than the reliability of that site. Thus it is our contention that ACC reports more nearly the availability as experienced by a user of the system.

3. Protocols

This section briefly describes each of the protocols studied in this paper. We divide the protocols into three classes, *static*, *dynamic*, and *oracular*. *Static* protocols, including the primary copy and the majority consensus protocols, use fixed criteria for assigning distinguished status to a component. *Dynamic* protocols, on the other hand, change these criteria based upon past events in the system. The majority of current protocol is a dynamic protocol. Lastly, we define *oracular* protocols to be those that change the criteria for distinguished status based upon both the past and the future events of the system. Although oracular protocols are inherently unimplementable in the real world, we will show them to be a useful tool for analysis. We refer to a protocol that does not make use of the knowledge of future events as a *non-oracular* or *implementable* protocol.

3.1. Primary Copy

The primary copy protocol[2] is the standard generalization of a non-replicated data object. Although more than one copy may exist in the net-

work, there exists a particular copy, designated the "primary", with which all accesses must communicate. This scheme is the simplest and easiest to implement, and it is against this scheme that replication protocols ultimately compete. In some networks, including certain networks studied in this paper, determining the optimal location of the primary copy is easy. However, finding the optimal location is in general NP-Complete[21]. In any event, we take this to be a seperate problem which, for the purposes of this paper, we assume to be solved.

3.2. Majority Consensus

The majority consensus protocol[20], MC, is an instance of the quorum consensus protocol[7], where no distinction is made between read and write operations. In the majority consensus protocol, each copy is assigned a number called its *vote assignment*, and a component is distinguished if and only if the total votes of all the copies in the component sum to a majority of the votes in the network. Since a majority of the votes is always required, it is clear that the consistency constraints are fulfilled.

The majority consensus protocol can be fine-tuned to maximize availability by varying the voting assignment. The best vote assignment will depend upon the topology of the network, the reliability of the components, and the distribution of the data accesses. Since in general finding the best vote assignment is computationally intractable, heuristics have been suggested to help choose an initial vote assignment[3, 17]. For this analysis, we use a uniform vote assignment of one vote per copy, since in our experiments the data access distribution and component reliabilities are all uniform and the topologies are roughly symmetric. An exception to this rationale will be discussed in section 5.3. By allocating to one copy a majority of the votes, majority consensus can achieve the same accessibility as the primary copy protocol, but this case must be selected before system startup and would not exhibit any benefits of replication.

3.3. Majority of Current

The majority of current protocol, MOC, is equivalent to an earlier protocol by Davčev and Burkhard[6] except that MOC does not require each site to possess instantaneous knowledge of all changes in network status. Two versions, identical for our purposes in functionality but differing in implementation, were presented by Jajodia and Mutchler[9, 12] and by Long and Pâris[18].

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

Two enhancements have been proposed since MOC was first introduced. The first is designed to choose, by assigning a linear ordering to the copies, a distinguished component from two components both containing half of the current copies[10]. In the case of such a tie, the component containing the current copy that is largest in this ordering becomes the distinguished component. We call this protocol majority of current with linear ordering, or $MOCLO$. The second enhancement is designed to increase availability by not allowing the distinguished component to shrink in size below two sites[11]. This is accomplished by reverting to the majority consensus protocol within a distinguished component of size three. In this paper, we analyze threshold values other than three and call this the MOC_i protocol, where i is the minimum size of a distinguished component. We call the combination of both enhancements the $MOCLO_i$ protocol. [1] It is apparent that neither enhancement violates the consistency constraints.

The majority of current protocols can also be generalized by weighting the copies using an initial voting assignment. For the same reasons given in section 3.2 above, we assign each copy a weight of one.

3.4. Oracle Protocol

The oracle protocol[8] calculates the sequence of distinguished components over time that maximizes the number of successful transactions for a given transaction stream. This is done off-line by finding the path of maximum cost in a graph representing the state of the network at the time of each

[1] We use an implementation different from that described in [11]. Instead of changing from the dynamic to the static phase by introducing component member-sets, we do so by simply freezing the version numbers. This method allows us to generalize i more easily, to retain the dynamic vote assignments in the static phase, and to decide whether a component is distinguished using the same test in both phases. Both implementations yield the same availability.

transaction submission.[2] Particularly interesting to us is that this simple concept provides such a tight upper bound to the actual performance of the realizable protocols. In this section we describe how to construct the graph, record the costs, and find the optimal sequence of distinguished components.

The oracle protocol has no a priori definition of a distinguished component. Instead the oracle uses the knowledge of future accesses to select the series of components that maximizes availability and calls those components distinguished. Obviously, the oracle is not allowed to violate the consistency constraints. As originally described[8] the oracle protocol required a uniform data access distribution and the knowledge of the failures and recoveries of network components. We modify the implementation here, requiring only the knowledge of each access and the state of the system at the time of each access, in order to more closely match the knowledge available to the realizable protocols and to allow for nonuniform transaction distributions.

The oracle can be implemented by creating a directed acyclic graph and finding a path of maximum cost. The graph consists vertices organized in levels, one level for each access, where every edge connects a vertex in level i (corresponding to the i^{th} access) to a vertex in level $i+1$. Each level contains one vertex for each component present in the system during the access creating this level. A down (non-operational) site forms its own component. An edge connects a vertex in level i to a vertex in level $i+1$ if and only if the corresponding components have at least one site with a copy in common. Each vertex is assigned a value of one if the access was presented to an operational site in the component represented by the vertex and the component contains a site with a copy. The vertex is assigned a value of zero otherwise.

After constructing such a graph, the oracle finds the path of maximum cost, where cost is defined as the sum of the values of the vertices along the path. The sequence of distinguished components over the entire series of accesses which will yield maximum availability is the sequence of components in order along the path of maximum cost. The oracle will deem these components distinguished and grant accesses accordingly. As with the realizable protocols, the oracle must abide by the consistency constraints. Since no edge connects two vertices of the same level and since the graph is acyclic, no two distinguished components can exist at one time. Since a

[2] If one is interested in the value of the maximum possible availability but not the path that yields this availability, then the calculation can be performed on-line using standard dynamic programming techniques. Also in this way, the oracle can be used in a real system as a measure of optimal system performance.

path in the graph must follow an edge from one level to the next, and since an edge exists if and only if the vertices connected by the edge have a site in common, successive distinguished components must have a site in common. Thus the consistency constraints are fulfilled.

4. System Model

Our desire to examine protocol performance in environments subject to component failures and consequent partitioning persuaded us to use simulation rather than the tools of mathematical analysis. Examining analytically a system of n sites and m links requires reducing the number of system states from 2^{n+m} to something more manageable, especially for $n = 101$ and m up to 5050 as in this paper. Unfortunately reducing the number of states in general requires some very strong assumptions such as a fully connected environment, infallible communication links, and updates between every failure and recovery event. Unwilling to make these assumptions, especially since they do not allow the partitioning which necessitates consistency control, we chose to simulate. An added benefit of simulation is the ability to calculate upper bounds on data availability as outlined in section 3.4.

We examine environments comprised of 101 sites configured into various topologies beginning with a ring, and adding links until all the sites are fully connected. We chose to consider ring-based networks since a ring is completely connected with the minimum number of links necessary to guarantee at least two disjoint paths between every pair of sites. Topology i is a ring with i additional links placed to maximize symmetry.

We study the performance of a single object under three scenarios. The first scenario, signified by $MOCLO$-1 and MC-1, is full replication. The second scenario contains five copies placed on five contiguous sites and is signified by $MOCLO$-2 and MC-2. The final scenario, signified by $MOCLO$-3 and MC-3, has five copies distributed to the five most highly-connected sites. Figure 4 shows the performance of the oracle protocol under these three scenarios and two similar scenarios using eleven copies. The performance of the primary copy protocol is independent of these scenarios since this protocol operates as if there is only one copy. All events, data accesses and component failures and recoveries, are modeled to occur instantaneously. Therefore a component can neither fail nor recover while an access is processing. All accesses are assumed to perform updates. This is equivalent to including both read and write accesses and maximizing write availability[7].

The submission of data access requests by each site is modeled as a Poisson process with mean μ_t. Site and link failures and recoveries are also modeled as Poisson processes. The mean time-to-next-failure of each component, μ_f, is the same for both sites and links. Likewise, the mean time to recovery, μ_r, is the same for both. Thus the *reliability* of a component, defined as the steady state probability that the component is operational, is $\frac{\mu_f}{\mu_f+\mu_r}$. In [5] Carroll and Long compare a number of component failure and recovery distributions and show that the choice of distribution does not significantly influence protocol performance. In addition, using the exponential distribution allows for the comparison between the results presented in this paper and those of analytic analyses requiring history independence[1, 3, 9, 10, 12, 14, 18, 15, 16].

Employing these three means as the model parameters allows us to treat a feature of the system which is critical to a proper interpretation. This feature is the ratio of mean time-to-next-access to the mean time-to-next-failure. We call this ratio ρ and begin the discussion of our results in section 5 by examining the effects of this parameter. For all our results we take $\mu_t = 1$. Also, we fix $\frac{\mu_f}{\mu_f+\mu_r} = .96$. Therefore all network components are 96% reliable. Our primary variable is ρ, and therefore μ_f and μ_r are dependent upon ρ.

In order to measure steady-state rather than finite-horizon performance it is necessary to avoid the effects of the initial, fully-operational state. We do so by refraining from statistic gathering until an initial number, I, of accesses and the events preceding these accesses have been processed. We studied the effects of various values of I and found that our results converged to the value obtained at I greater than $50,000$. We chose $I = 100,000$ for the simulations reported in this paper, which we find to be sufficient to overcome the effects of the initial state.

We followed a similar procedure to determine a finite horizon that approximated steady-state. Here we found that any number of accesses in excess of $100,000$ yielded the same performance. Each availability figure reported in this paper reflects the average availability over a number of *batches* each consisting of $1,000,000$ accesses. A *batch* is a series of events, and the number of batches is dictated by the desired confidence interval. The network is reset to the initial state before each batch is begun.

All simulations were run on a DEC 5000 and took between one and three hours depending upon the topology and value of ρ. All protocols and copy

placement schemes were run together on the same series of events. Availabilities reported are with a 95% confidence interval of $\pm 0.5\%$.

5. Simulation Results

5.1. Influence of ρ

We begin by noting the significant influence of the value of ρ, the ratio of the mean time-to-next-access to the mean time-to-next-failure. Figure 1 shows that as ρ decreases, resulting in fewer failures each of a longer duration since component reliability is held constant at 96%, the performance of $MOCLO$ increases quite substantially. The most dramatic difference occurs using topology 1 where the availability of $MOCLO$ reported at $\rho = \frac{1}{1}$ is less than eighty percent of that reported at $\rho = \frac{1}{512}$. We choose to concentrate on $\rho = \frac{1}{128}$, since this ratio is low enough to be realistic yet high enough to display visible distinctions between the performances of the protocols.

Like the performance metrics discussed in section 2, ρ affects the protocols disproportionately. The extent to which the network state may change between accesses is governed by ρ. When ρ is large, the network may change drastically between accesses. As ρ decreases, the probability of drastic change also decreases. Thus the ability of a dynamic protocol to ensure, as required by consistency constraint 2, that consecutive distinguished components overlap increases as ρ decreases. Unlike the dynamic protocols, the static protocols are uneffected by changes in the value of ρ when component reliability is held constant. Since the criteria for distinguished status can not change, consistency constraint 2 requires that every two possible distinguished components have a copy in common. Thus the static protocols are independent of the commonality between successive network states. Finally, ρ affects the oracular protocols in the same way that it effects the dynamic protocols.

5.2. Performance of MOC Enhancements

In this section we examine the two enhancements suggested for the majority of current protocol as described in section 3.3. Tables 1 and 2 show the performance of each enhancement as a percent availability. Since $MOCLO_i$ performed better than MOC_i for all configuration that we tested, we do not include statistics for MOC_i.

We see from Tables 1 and 2 that the performance of linear ordering always surpasses that of MOC. This seems reasonable since it makes no

topology	*moc*	*moclo*	$moclo_3$	$moclo_6$	$moclo_{12}$	$moclo_{24}$
0	26.2	26.9	26.9	26.9	26.4	22.8
1	40.7	41.2	41.2	41.2	41.0	38.3
2	58.8	51.2	51.2	51.2	51.0	49.3
4	64.7	65.0	65.0	65.0	64.9	64.1
16	85.6	85.7	85.7	85.7	85.7	85.7
256	96.0	96.0	96.0	96.0	96.0	96.0
Full	96.0	96.0	96.0	96.0	96.0	96.0

Table 1: Performance of *MOC* Enhancements
Percent Availability
(fully-replicated)

	copies distributed			copies adjacent		
topology	*moc*	*moclo*	$moclo_3$	*moc*	*moclo*	$moclo_3$
0	13.1	24.9	24.9	23.3	23.4	23.4
1	37.1	42.4	42.3	42.9	43.0	43.0
2	43.9	50.0	50.0	47.3	47.6	47.6
4	59.7	63.4	63.4	59.3	59.7	59.6
16	83.5	84.6	84.6	82.1	82.5	82.5
256	96.0	96.0	96.0	96.0	96.0	96.0
Full	96.0	96.0	96.0	96.0	96.0	96.0

Table 2: Performance of *MOC* Enhancements
Percent Availability
(five copies)

sense in general to refuse distinguished status to both components merely because they have equal vote totals. Equally as obvious is the observation that when the data object is fully-replicated the hybrid enhancement does not prove beneficial under any circumstances. This may be explained as follows: For large networks, components of size six or less are unlikely to form and are even less likely to be distinguished under the $MOCLO$ criteria. In addition, these components tend to grow rather than subdivide further, and thus a low threshold has no effect. For thresholds of size greater than six, the dynamic nature of $MOCLO$ and its associated advantages over the static protocols are defeated.

Table 2 shows that this enhancement has virtually no effect on networks with five copies. $MOCLO_3$ differs from $MOCLO$ only when a distinguished component containing only one copy is attempted immediately after a distinguished component containing two copies is allowed. This single copy distinguished component is allowed by $MOCLO$ but is forbidden by $MOCLO_3$. $MOCLO_3$ performs better than $MOCLO$ only if this distinguished component containing only one copy is then isolated from the rest of the network for a duration of time long enough to compensate for this lost opportunity. This sequence of events seems improbable in a large network of highly-reliable components. The superiority of this hybrid method for systems of five sites and infallible links as reported by Jajodia and Mutchler[11] is likely to be limited to very small networks.

5.3. Performance of Majority of Current

In Figure 2 we compare the performance of $MOCLO$ to that of the primary copy protocol and to the oracle upper bound. We show $MOCLO$ under three different conditions:

1. $MOCLO$-1 with full-replication.
2. $MOCLO$-2 with five copies at five consecutive sites.
3. $MOCLO$-3 with five copies distributed to the most highly-connected sites in the network.

From the figure we see that $MOCLO$-1 performs at least as well as the other realizable protocols for all topologies but one. The single instance where the performance of the primary copy protocol exceeds that of $MOCLO$ is the fault of the vote assignment rather than the $MOCLO$ protocol. In section 3.2 we defend the use of a uniform vote assignment by asserting the relatively

symmetric nature of the topologies. This is not the case for topology 1, since topology 1 is a ring along with an additional link from site zero to site fifty. By assigning a greater number of votes to site zero or site fifty or both, we can increase the performance of *MOCLO* to at least that of the primary copy protocol. The difficulty, as mentioned earlier, lies in determining an optimal vote assignment for a given topology and event distribution.

Not surprisingly, Figure 2 shows that *MOCLO* performs better when data is fully-replicated than when there are only five copies. The interesting question is whether this difference is due to a change in the potential availability of the system (as measured by the oracle protocol) or is solely a characteristic of the majority of current protocol. Figure 4 shows that the difference is not due to an increase in the potential of the system, since the performance of the oracle protocol changes little when replication is increased from five distributed copies to full replication. Instead, the heuristics of *MOCLO* perform better in fully replicated systems than in five-copy systems.

On the other hand, Figure 4 shows that the decline in *MOCLO* performance given five copies located on adjacent sites is largely due to a decrease in system potential. Figures 2-4 show that all the protocols, including the oracle and the primary copy protocol, perform nearly identically when five copies are placed on adjacent sites. The only exception to this is for the highly-connected topologies 256 and 4949 as discussed in section 5.5. This implies that availability is maximized when copies are distributed uniformly about the network rather than placed on adjacent sites.

An important implication of Figure 2 is that under full replication *MOCLO* performs within six percentage points of optimal for all topologies and performs optimally for topologies 16, 256, and 4949. As ρ decreases (cross-reference Figure 1), the performance of *MOCLO* relative to the oracle continues to improve. We conclude that under this model there is little room for the development of a protocol with performance superior to that of *MOCLO*. The only exception may be improved performance in very low connectivity networks (i.e. the ring and topology 1), but the availability in these networks is so low that they would be undesirable even at optimal performance.

5.4. Performance of Replication

Turning our attention to replication in general, we see that although the oracle and *MOCLO* perform the best, the primary copy protocol frequently

does nearly as well. As Figure 2 indicates, the average performance advantage of $MOCLO$-1 over the primary copy protocol is only 2.9 percentage points, and the maximum advantage is 5.9 percentage points. At this point of maximum difference, the primary copy protocol yields availability 59.11% and $MOCLO$-1 yields availability 65.01%. This advantage decreases as the level of replication decreases. In many applications, the increases in performance due to replication may be too small to warrant the additional hardware and software complexity and communication costs incurred by replication.

5.5. Performance on Highly-Connected Topologies

Interestingly, the availability results for highly-connected networks can easily be predicted. We see from Figures 1-3 that the availability resulting from MC and MOC protocols is 96% and the availability resulting from the primary copy protocol is 92%. Not coincidentally, this is p and p^2, respectively, where $p = \frac{\mu_f}{\mu_f + \mu_r}$, the reliability of the sites and links. We show why this is true as follows: suppose that only site i has a copy of the data object and that an access request is submitted at site j, $i \neq j$. Using the primary copy protocol, the probability that site j can access the data item is the product of the probabilities that site j is up, that site j can communicate with site i, and that site i is up. Now for a highly-connected network of n sites, the number of paths from j to i is very large, and therefore the probability that site j can communicate with site i for $n = 101$ is, for all practical purposes, equal to one. Thus the probability that j can access the data object is p^2.

If, on the other hand, we replicate the data object at m sites and if, like MC, we require that a certain number r of these copies be accessible to site j, then the accessibility approaches p as the number of sites increases. Just as the number of paths gets large in the number of sites, so too does the number of cardinality r or greater subsets of the m copies. More precisely, the probability that at least r of the m copies are all on operational sites is $\sum_{k=r}^{m} \binom{m}{k} p^k (1-p)^{m-k}$. For $m = 5$, $r = 3$, $p = .96$, this is .9994. For $m = 101$, $r = 51$, $p = .96$, this is, for all practical purposes, equal to one. As above, the probability that site j can reach at least r of these sites is also nearly one due to the high number of paths. Thus we are left only with p, the probability that site j is operational.

6. Concluding Remarks

We draw three important conclusions from this work. Firstly, the interpretation of the availability results is substantially dependent upon not only the accesses request rate and component reliability, but also upon the ratio of accesses to component failures (ρ). We have also shown that for the ACC (Accessibility) availability measure and reasonable ratios of accesses to component failures ($\rho \leq \frac{1}{128}$), the majority of current protocol on large networks performs nearly optimally. As a matter of fact, there is little room for enhancements beyond the linear ordering mechanism for highly-connected topologies. On the other hand, the primary copy protocol is inferior to $MOCLO$ (majority of current using full replication) by at most 5.9 percentage points for these same topologies and is inferior by less for lower levels of replication. This indicates that replication may not be as helpful as previously thought. For highly-connected topologies, we know that the performance of a single copy will be inferior to that of a fully-replicated system by a factor of p, the reliability of a site.

Before deducing that replication is of little value in increasing data availability, we must examine the assumptions on which our analysis rests. The assumption of uniform access distribution and update-only accesses may fail to demonstrate the full potential of replication in real systems, while the assumption of single-object accesses may lead to overstating the value of replication.

The assumption of a uniform access distribution implies that the location of the next access request is independent of the location of the current access request. If this were not the case, then there would be the possibility of locality in the object reference pattern. A stationary primary copy scheme can not take advantage of such a shifting pattern of data access, whereas it may be possible to exploit locality using a dynamic protocol.

The influence of the assumption of only update accesses is also in favor of a single copy protocol. If read accesses are allowed, then a dynamic protocol like MC and MOC can optimize for reads by reducing the number of votes that a site must collect in order to allow a read access to a data item. This number of votes is called the read *quorum*. Unfortunately, reducing the size of a read quorum requires a corresponding increase in the size of a write quorum in order to ensure data consistency[7]. Thus the increase of read availability is at the expense of write availability. The exact nature of this trade-off is a function of, among other things, the read-write mix of the application under consideration. As we stated earlier, the availability

of update accesses is equivalent to the availability of both read and write accesses where the read quorum equals the write quorum, and therefore maximizes write availability.

Lastly, we have studied the availability of a single object, whereas the objective of a system manager is to maximize the probability of the success of an entire transaction. These two availabilities are identical if the data base is replicated as a whole, rather than replicating each object separately. If, on the other hand, each object in the data base is replicated independently, then the availability of a transaction may be much worse than that of any single object. The actual availability of the transaction is a function of the network paths taken to the data objects. The more independent the paths, the lower the availability of the transaction.

Analyzing the effects of read accesses, nonuniform access submit distributions, and multiple accesses per transaction is an important area of further work. This paper has shown that the benefits of replication under the standard assumptions of uniform access distribution and update-only transactions are slight, and that benefit can be realized with existing consistency control protocols.

References.

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

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

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

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

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

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

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

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

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

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

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

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

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

[14] Darrel D. E. Long, John Carroll, and Kris Stewart. Estimating the reliability of regeneration-based replica control protocols. *IEEE Transactions on Computers*, 38(12):1691–1702, December 1989.

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

[16] Darrell D. E. Long. *The Management of Replication in a Distributed System.* PhD thesis, University of California at San Diego, 1988.

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

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

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

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

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

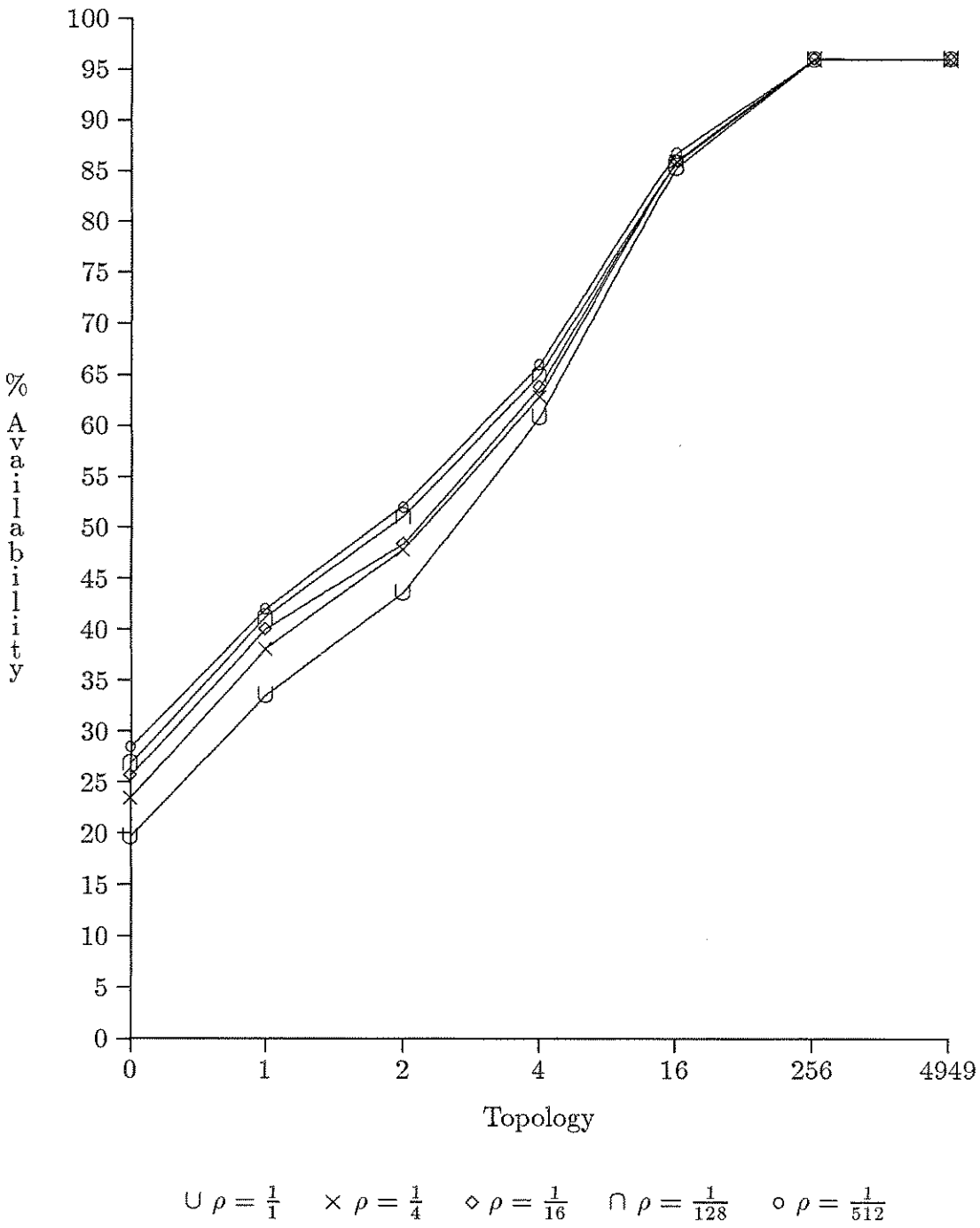


Figure 1: Influence of ρ on MOCLO Performance
(full replication)

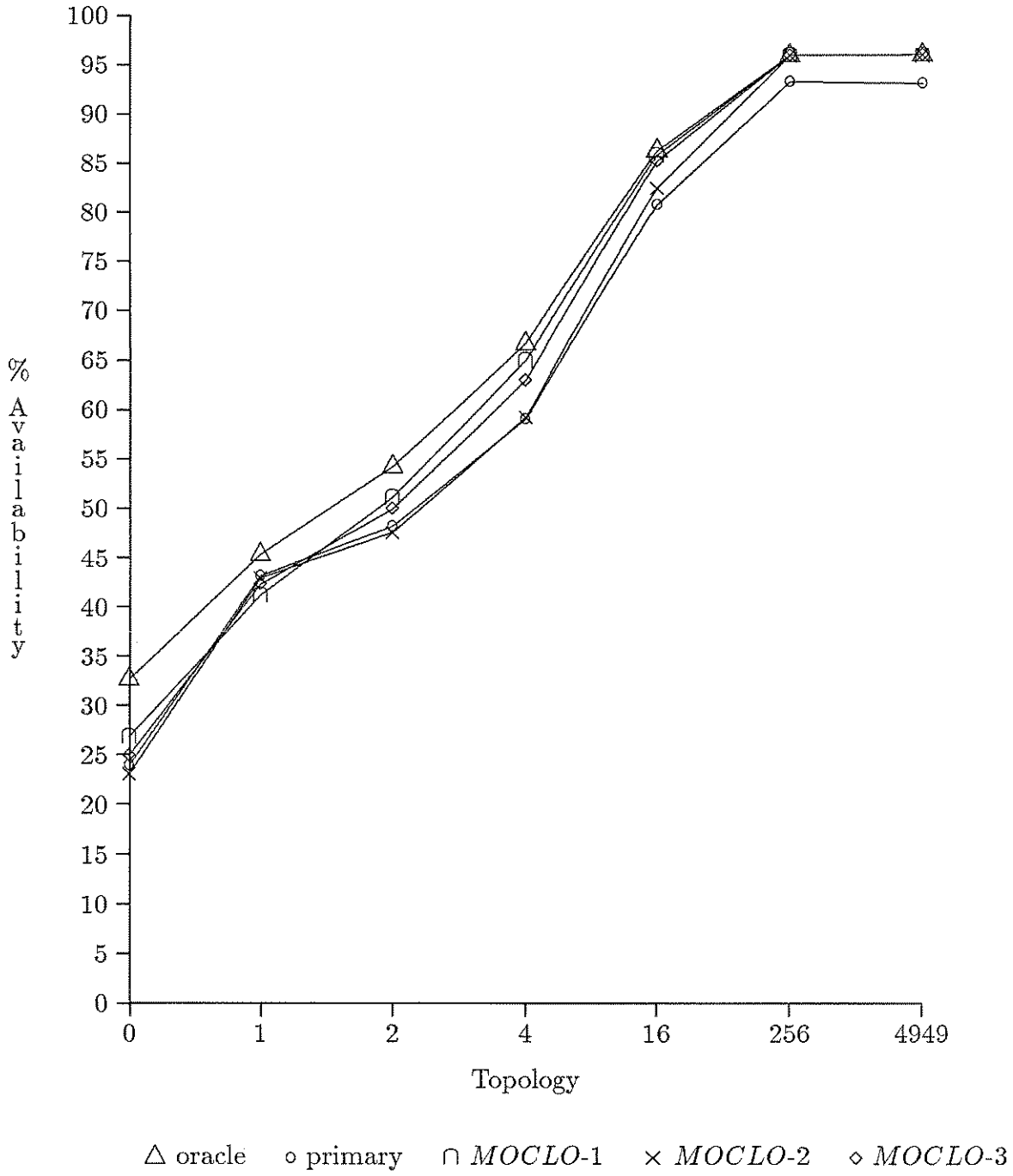


Figure 2: Performance of MOCLO
($\rho = \frac{1}{128}$)

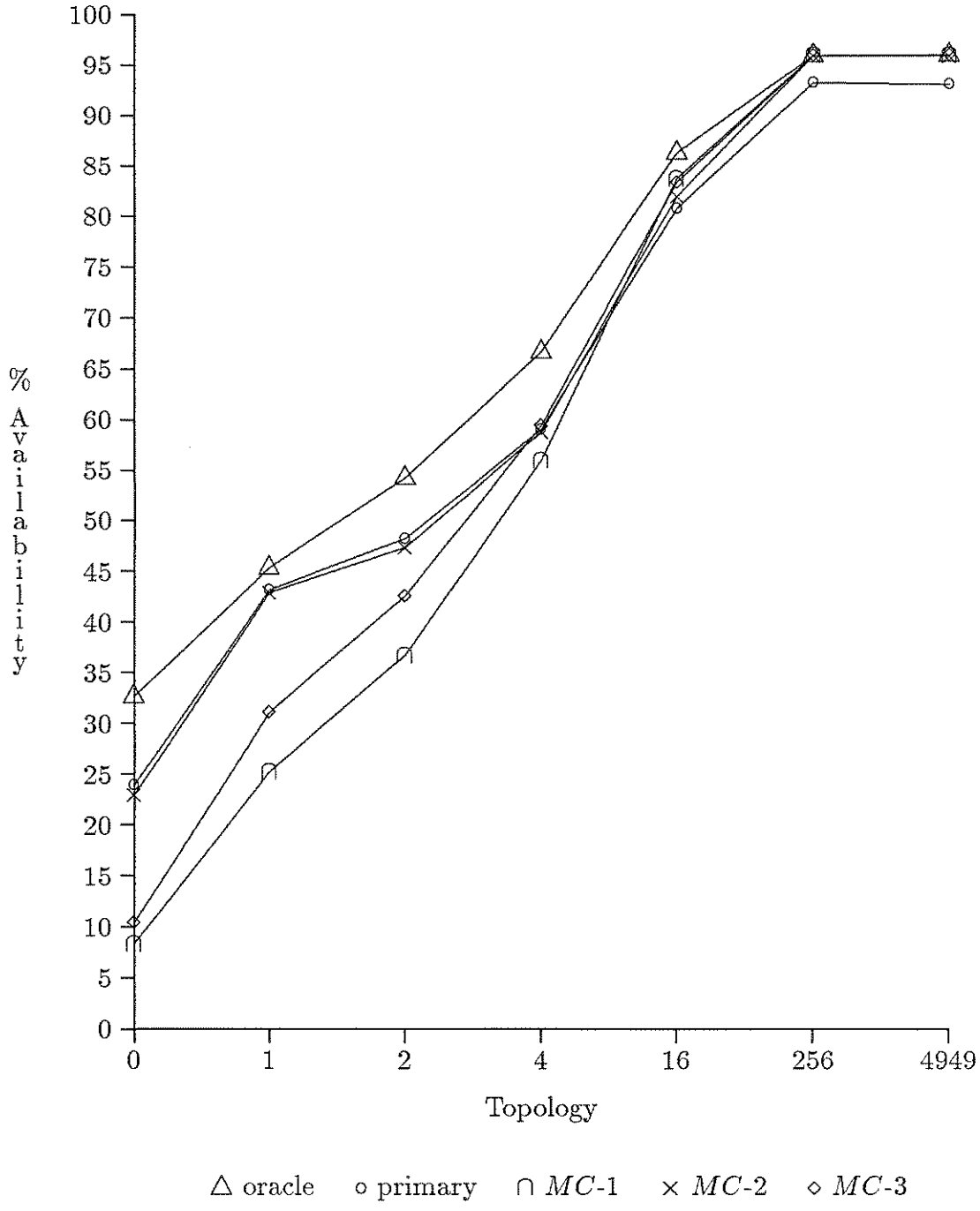
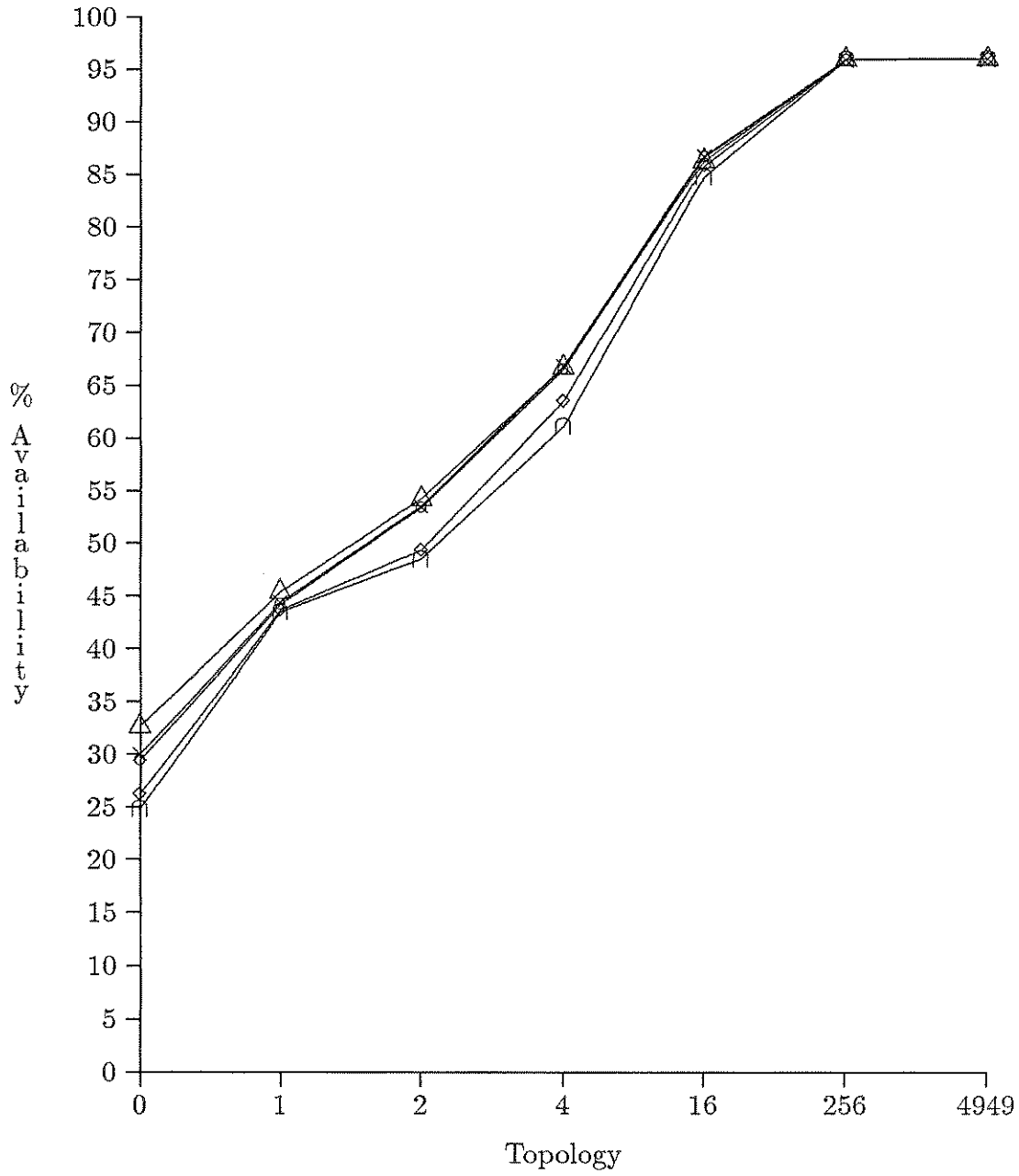


Figure 3: Performance of MC
($\rho = \frac{1}{128}$)



symbol, number of copies, distribution of copies:

△ 101 ∘ 5 separate ∩ 5 together × 11 separate ◇ 11 together

Figure 4: Performance of Oracle
($\rho = \frac{1}{128}$)