7-26-1989

A Comparison of Consistency Control Protocols

Michael Goldweber
Dartmouth College

Donald B. Johnson
Dartmouth College

Larry Raab
Dartmouth College

A COMPARISON OF CONSISTENCY CONTROL PROTOCOLS

Michael Goldweber, Donald B. Johnson,
Larry Raab

Technical Report PCS-TR89-142

A Comparison of Consistency Control Protocols

Michael Goldweber* Donald B. Johnson†
Larry Raab‡
Dartmouth College§

July 26, 1989

Abstract

In this paper we analyze three protocols for maintaining the mutual consistency of replicated objects in a distributed computing environment and compare their performance with that of an oracle protocol whose performance is optimal. We examine these protocols, two dynamic protocols and the majority consensus protocol, via simulation using two measures of availability. The analysis shows that the dynamic protocols, under realistic assumptions, do not perform significantly better than the static voting scheme. Finally we demonstrate that none of these approaches perform as well as our oracle protocol which is shown to be an upper bound on availability.

1 Introduction

In a distributed system, data replication can be used to improve both speed of data access and availability of data objects. Accessing a local copy of a

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

replicated object eliminates the need for time consuming communication. By storing copies of an item on more than one processor, the probability that at least one copy is accessible increases.

The full benefits of data replication are in practice very difficult to achieve because of the need for data *correctness*. 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*[5]. 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 provide one-copy serializability can be classified as either optimistic or pessimistic[6]. An optimistic strategy maximizes data availability by allowing a transaction to execute in any partition containing at least one accessible copy of all objects referenced. Although each block runs a local concurrency control function to insure the serializability of transactions committed within the block, the global set of committed transaction may not be serializable. As a result, the protocol must detect and resolve these inconsistencies by backing out some subset of committed transactions. Optimistic protocols perform well when the probability of inter-partition conflict is low; a large number of objects, a high percentage of read-only transactions, and a low probability of network partitioning.

The protocols that we examine are all pessimistic strategies, which means that mutual consistency is strictly enforced by restricting availability, thereby guaranteeing that committed transactions are never backed out. *Mutual exclusion*, the mechanism used by these strategies to guarantee the consistency requirement, allows at most one set of communicating sites to access a replicated object at a time. This set of communicating sites is called the *majority partition*.

The system we consider is composed of sites and bi-directional links. Links fail by crashing or by delaying or 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[14]; 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. 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 description of our system model and performance metrics. Then we briefly describe the protocols that we analyze which include the majority consensus protocol[16] (including primary site [1,15]), the family of vote reassignment protocols by Barbara et al. [4,3], and the enhanced dynamic voting protocol by Jajodia and Mutchler[12] (including Davčev and Burkhard's dynamic voting scheme[7]). We then introduce the oracle protocol used to provide an upper bound on data availability. We describe our simulator and its parameters and assumptions. Lastly we analyze our simulation results and compare these results to those of [4] and [12].

2 Measures of Data Availability

There are two definitions of data availability in the literature[11]. The first, which we will call the SIM metric, is the probability at time t, as t goes to infinity, that an arbitrary site is a member of the majority partition. The second metric, which we will refer to as OPT, is the probability at time t, as t goes to infinity, that there exists a majority partition for a given data object.

We have chosen to present our results using the SIM metric rather than the more common OPT metric since it more accurately reflects what we consider to be the intuitive, relevant definition of availability. Although OPT recognizes the necessity of the existence of a majority partition for the success of a transaction, it fails to capture the influence of the size of that partition on the probability that a particular transaction at a particular site will succeed. In order for the OPT metric to be a practical measure of this probability, it would be necessary for a transaction to possess the knowledge of where to submit itself. Such knowledge is clearly not available

to a transaction under our system model.[1]

Although this distinction does not change the performance ranking of the protocols relative to each other, it does influence the extent to which one protocol out-performs another. In particular the OPT metric, by neglecting to consider the size of the majority partition, accentuates the advantage of protocols such as the dynamic protocols that allow smaller majority partitions. To a large extent, this explains why our results will show a significantly smaller advantage of the dynamic protocols over majority consensus than results previously reported[3,12].

If transactions are distributed randomly throughout the network and their arrival rate is exponentially distributed, then one can quantify the relationship between the SIM and OPT measures of availability. Let $p(s)$ be the percentage of time there was a majority partition of size s, where s ranges from zero (no majority partition) to n, the number of sites in the system. We found that:

$$OPT = \sum_{s=1}^{n} p(s)$$

$$SIM = \sum_{s=1}^{n} \frac{s}{n} p(s)$$

Figure 1 illustrates the difference between these two metrics using our oracle protocol which measures the optimal percentage of successful transactions.[2] From this figure we see that the OPT metric yields greater availability then the underlying network, thus a submitted transaction has a greater probability of succeeding than the probability that a random site is operational. Clearly the assumptions necessary for this to be true (transactions knowing where to be submitted and having the ability to be submitted at the chosen site) are unrealistic, and therefore they are not included in our model of transaction processing.

[1] Although similar reasoning is used by Jajodia and Mutchler to justify their use of the SIM metric in their first paper[11], they switch to the OPT metric with little explanation in their second paper[12].

[2] This oracle protocol is described fully in section 4.

3 Protocols

In addition to the standard static voting protocol we examine two dynamic approaches. A dynamic protocol is one that alters the requirements for choosing a majority partition as events change the network configuration. These protocols attempt to increase the probability that, after the next component failure, a majority partition still exists.

3.1 Static Voting

In the quorum consensus protocol[10], each copy of a replicated object is assigned a fixed number of votes. Every transaction must acquire a read quorum of votes $r[x]$ for an objects x in its readset, and a write quorum of votes $w[y]$ for an objects y in its writeset. The read and write quorum values are assigned a priori and must satisfy the following conditions:

- $r[x] + w[x] > n[x]$ where $n[x]$ is the total number of votes assigned to object x.
- $w[x] > n[x]/2$.

These conditions insure there exists a non-empty intersection between every read and write quorum and that updates cannot occur in more than one partition. These conditions are also sufficient to guarantee that conflicts cannot occur between transactions executing in different blocks. Majority consensus[16] is a special case of quorum consensus in that under majority consensus $r[x] = w[x] = \lceil (n[x] + 1)/2 \rceil$.

Garcia-Molina and Barbara have shown that not all valid majority consensus quorum assignments can be represented by a weighted voting[2,8] and therefore proposed coteries as a method of specifying all valid quorum groupings.[3] While any majority consensus vote assignment can be expressed as a coterie, not every coterie can be expressed as a majority consensus vote assignment implying coteries is the more complete specification of quorum assignments.

The quorum consensus protocol can be fine-tuned to reflect a desired availability by varying both the initial vote assignments[9] and the read

[3]Coteries make no distinction between reads and writes implying $r[x] = w[x]$.

and write quorums. Examples of this include assigning votes to copies proportional to the reliability of the node upon which they are stored or choosing $r[x] < n[x]/2$, thereby allowing the possibility of an object to be read in more than one partition (in which case no partition would be capable of accumulating $w[x]$). One can achieve the same accessibility as the primary copy protocol[1,15] by allocating to one copy a single vote and to all other copies zero votes. Since the system we examine is comprised only of update transactions we chose to minimize the size of the write quorum, maximizing availability. This was done by setting $r[x] = w[x]$ and varying only the initial vote assignments.

3.2 Vote Reassignment

The dynamic approach of Barbara et al.[4] is a family of protocols that dynamically reassigns votes upon site and link failures. The scheme is comprised of two functions, one for vote reassignment and the other for vote collection. The vote reassignment function allows a copy to autonomously increase its vote assignment (assuming it is a member of the majority block) by broadcasting its new vote value. The vote collection function both determines whether a partition is a majority partition and propagates new vote values throughout the partition. The scheme guarantees mutual exclusion since at most one partition at any time will consider itself the majority for any given data object.

There are several polices for deciding which copy should alter its votes and by what amount[3]. Either everyone in the majority partition increases its vote (alliance techniques) or only one copy increases its vote (overthrow techniques). Other policy choices determine the amount of increase and the actions taken upon site recoveries and partition merges. The other policy choices involve the quantity of the increase and what to do when a site recovers or partitions merge. During recovery either the newly joined copy can increase its vote to catchup with the rest or the old members can decrease their votes, thus approximating the original distribution of voting strength. A common characteristic of these polices is that they will after a site failure or network partition attempt to redistribute two times the "lost" votes among the remaining members of the majority partition (assuming a majority still exists) so as to be resilient to the next failure. Incrementing the vote total by this amount covers not only the lost votes but also the

increase in total votes, thus any majority that could be formed with the lost site(s) can still form using the site(s) whose votes were increased. We examine the alliance technique using both the catchup and decrease strategies, assigning each member of the majority twice the lost votes and the overthrow scheme with the decrease option only.[4]

3.3 Majority of Current

The Majority of Current (MOC) protocol is a read-one/write-all scheme that defines the majority partition as the physical partition containing a majority of the current copies[12]. Associated with each copy i of data item A are the vote value (VT_i), version number (VN_i), update sites cardinality (SC_i) (which reflects the total number of votes participating in the most recent update to copy i) and the distinguished copy variable DS_i which identifies the greatest numbered copy from an a priori numbering on all copies of A that participated in the most recent update to copy i. A transaction collects these four values from each of the m accessible copies (denoted $A_1, \ldots, A_m$) and calculates:

- the value $M = \max(VN_i : 1 \leq i \leq m)$
- the set $I = (A_j : VN_j = M,\ 1 \leq j \leq m)$
- the value $N = \max(SC_k : A_k \in I)$
- the value $VOTES = \sum VT_k : A_k \in I)$

If $VOTES > N/2$ or both $VOTES = N/2$ and $DS_i \in I$ where DS_i is the distinguished copy value from any member in the set I, then the transaction can claim a majority with respect to A. VN_i, SC_i, and DS_i are updated every time A is written to (at transaction commit), while the vote assignment VT_i always remains the same. Actually DS_i need only be reset when N is even.

The inclusion of DS_i is an enhancement over the original majority of current protocol[11] (called dynamic voting by its authors) by not forcing a halted state when N is even. The dynamic voting protocol is itself an

[4]The overthrow technique with the catchup option would not help to maintain the original distribution of voting strength.

implementable variation of the version vector protocol[7] yielding the identical level of data accessibility without being dependent upon unrealizable assumptions.[5]

4 Oracle Protocol

Given the consistency constraint, the problem of determining the upper bound on availability for a particular series of link and node failures and recoveries, collectively called *events*, can be stated as follows: Given a series of events and the times at which they occur, what is the assignment of majority partitions that maximizes availability? Viewed in this manner, a pessimistic consistency protocol is simply a heuristic for deciding, after each event, which partition should be given access to each data item. The two constraints relevant to this decision for each object are (1) the set of copies in two successive majority partitions must have at least one copy in common, and (2) no two majority partitions may exist simultaneously.

In order to obtain on upper bound on availability, we designed an oracle protocol. After each event, the oracle is queried with the question "which of the currently existing partitions should be designated the majority partition?" The oracle answers in such a way as to abide by constraints (1) and (2) above and maximize availability over the entire series of queries, including future queries.

The oracle can be implemented by creating a graph and finding a path of maximum cost. The directed acyclic graph $G = (V, E)$ consists of levels, one level for each event, where every edge connects a vertex in level i to a vertex in level $i + 1$. Each level contains one vertex for each partition present in the system after the event creating this level. The *time* of a level is the time at which the event forming the level occurred. For $v_1, v_2 \in V$, $(v_1, v_2) \in E \; iff$ the partitions v_1 and v_2 have at least one copy in common

[5] In the version vectors protocol each site contains a symmetric and transitive P-bit connection vector indicating with which of the P sites it can communicate. The protocol requires that changes in the system configuration from site/link failures and recoveries are instantaneously recorded in the proper connection vectors. Furthermore the recovery/merge algorithm as described does not guarantee mutual exclusion (leading to non-serializable executions). When a site/block merges with the majority it should adopt the majority's complete version vector (with the appropriate new zero entries) as well as the current value of the data object.

and the level containing v_1 immediately proceeds the level containing v_2. The cost on an edge (v_1, v_2) is the number of sites in v_1 multiplied by the time between the levels containing v_1 and v_2. If v_1 contains a down site (in which case the partition v_1 contains only this one site) the edge cost is zero. G also contains two distinguished vertices s and t. E contains an edge (s, v_i) for all v_i in the first level that contain a copy and an edge (v_j, t) for all v_j in the last level. The edge cost for (s, v_i) is the number of sites in the system multiplied by the time of the first event. If the system is partitioned at startup, we add that partitioning as tbe first event and assign it a time of zero. The edge cost for (v_j, t) is the number of sites in v_j multiplied by the time of the last level to the end of the simulation.

After constructing such a graph and finding the path of maximum cost, the oracle knows the sequence of majority partitions over the entire simulation which will yield maximum availability. This sequence is the sequence of partitions in order along the path, and the oracle will answer queries accordingly. The maximum percentage of availability as defined by the SIM metric can be found by dividing the cost of the maximum path from s to t by both the number of sites in the system and the total time of the simulation.

5 System Model

The decision to use simulation rather than an analytic model as the method of analysis was dictated primarily by the weakness of our assumptions. Under these assumptions, a system of m sites and n links has 2^{m+n} possible states. Previous analytic analysis[12] characterized a system in three variables by combining a number of system states into one state in the model. This reduced the 2^{m+n} possible system states to $4m - 2$ states. Unfortunately, this required a number of very strong assumptions including a fully connected environment, infallible communication links, and updates between every failure and recovery. Unwilling to make these assumptions, especially since they do not allow the partitioning which necessitates consistency control, we were presented with a system that could only be characterized with an exponential number of states.

We examine environments comprised of four, five, six, and seven sites configured into various topologies beginning with a ring, and adding links

until all the sites are fully connected.[6] Both links and sites are modeled as Poisson processes with the mean time to failure of a functioning component being $1/\lambda$ and the mean time to repair of a failed component being $1/\mu$. Each topology was examined under four scenarios:

- $\lambda = 2\mu$ (33% node/link operating probability)
- $\lambda = \mu$ (50% node/link operating probability)
- $\lambda = .5\mu$ (66% node/link operating probability)
- $\lambda = .25\mu$ (80% node/link operating probability)

We assume a single object, fully replicated environment. Transactions, whose arrival rate is exponentially distributed around a mean of one, are randomly submitted throughout the system and always perform updates. All events are modeled to occur instantaneously, therefore a component can neither fail nor recover while a transaction is processing. Each protocol is examined using different initial vote assignments. Finally, all non-dominated vote assignments[7] are simulated for the four and five site cases. The six and seven site cases are analyzed using the primary site and non-dominated equal distribution vote assignments.

6 Simulation Results

Figures 2-5[8] compare the performance of each protocol under the SIM measure of availability using seven sites[9] and various component reliability rates. One can observe that no protocol that we examined achieves the optimal percentage of successful transactions as indicated by the Oracle. On the other hand, as reliablility increases these protocols' performance steadily approaches that of the Oracle's. Once we reach eighty percent

[6] Similar to the approach taken in [3].

[7] d_1 is a dominated assignment iff there exists another assignment d_2 such that each majority partition capable of forming under assignment d_1 can also form under d_2 and there exists a majority partition that can form under d_2 but not under d_1[2,8].

[8] All results obtained a 95% confidence interval $\pm$ 3%.

[9] All results reported for seven sites also apply for four, five, and six sites unless otherwise noted.

component reliability and high levels of network connectivity, the best of the protocols exhibit nearly optimal performance.[10] From these graphs one also observes that the protocol which performs best is the majority of current protocol. Although the static and vote reassignment protocols frequently perform as well, no protocol ever exceeds the performance of majority of current. This is contrary to an "aberrant" case reported by Jajodia and Mutchler where static voting is superior to their protocol for five sites and reliabilities less than 50%[11]. We find our uniform results more convincing than the explanations given by Jajodia and Mutchler for the existence of the behavior they report.[11]

Figures 6-9 illustrate a principle which can simplify the task of finding the best static vote assignment. For both four and five sites, the vote assignment that yields the greatest availability is either the primary site assignment or the uniform assignment which gives all sites one vote.[12] In general, the primary site assignment is best for low reliabilities, and the uniform assignment is superior for high reliabilities. Since all valid majority consensus quorum assignments for five or fewer sites can be characterized by a weighted voting, these results generalize to coteries. We also suspect that these results generalize to six or more sites, but since we know of no efficient algorithm for computing all the non-dominated weighted vote assignments much less all valid non-dominated majority consensus quorum assignments, we are unable to exhaustively test this conjecture. We do know, however, that the quorum assignment given in the literature as an example of a non-dominated coterie with no equivalent voting assignment[8] is at each reliability inferior to either the primary site or uniform assignment.

Although the choice of an initial vote assignment can be simplified by considering only the primary copy and uniform assignments, it is an advan-

[10]A point on these graphs indicates the *best* performance of the indicated protocol where *best* is defined to be the protocol's performance using the initial vote assignment which yields the highest percentage of successful transactions for that topology. Since two colinear points may not represent the same initial vote assignment, there may be no combination of protocol and vote assignment that achieves the performance represented by an entire line.

[11]This behavior is explained in the first paper as "a residual effect from the 3-site subsystem"[11]. The authors assert that this behavior disappears in their second paper as a consequence of switching from the SIM to the OPT metric[12].

[12]For an even number of sites, exactly one site is given two votes yielding an odd number of total votes, and therefore guaranteeing a non-dominated voting assignment.[8]

tage of the vote reassignment protocols that their performance is virtually independent of the initial vote assignment. This advantage may be valuable in a highly dynamic system where the reliability fluctuates so severely that no one static vote assignment is optimal over the entire range of reliabilities. On the other hand, when reliability is relatively stable, figures 2-5 show that there always exists a static vote assignment that performs at least as well as any of the vote reassignment protocols. This result seems to contradict the performance evaluation as reported in [3] where the Barbara et al. show their protocols significantly superior to the static protocol. Two differences between their simulation and our own account for this difference. Firstly, as discussed earlier their use the OPT metric accentuates the advantage of such a dynamic protocol over a static protocol. The second reason why we do not show any advantage of the vote reassignment protocols is that the initial vote assignments chosen by Barbara et al.[13] are not equivalent to those we have shown to be optimal for the static voting protocol under the conditions tested ($\lambda = \mu$). Casting our results in the OPT metric, we too show the vote reassignment protocols superior to the static protocol (see Figure 10) for those vote assignments used by Barbara et al..

7 Conclusions

We have demonstrated that under a realistic measure of availability the performance of the protocols we have studied varies considerably with the reliability and topology of the system. For systems with low component reliability or sparse connectivity, there is a significant gap between the performance of the best protocol and the optimal performance as measured by the Oracle. On the other hand, these same protocols operate very close to optimal under higher reliabilities and connectivities.

We have also shown that of the protocols we have studied, the majority of current protocol performs the best over all reliabilities and connectivities. Surprisingly, the next best protocol, the static voting protocol, performs nearly as well. This is important since the static protocol is the easiest of the three protocols to implement, especially since one need only consider the primary-copy and the uniform voting assignments. While the majority

[13]The assignments in [3] are equivalent to our uniform vote assignment vote6, except for topology 1 where their assignment is equivalent to our static vote 5.

of current protocol is not significantly more difficult to implement, the static protocol requires lower communication and storage overhead and has a relatively simple correctness proof.

References

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

[2] Daniel Barbara and Hector Garcia-Molina. Mutual exclusion in partitioned distributed systems. *Distributed Computing*, 1:119–132, 1986.

[3] Daniel Barbara, Hector Garcia-Molina, and Annemarie Spauster. Policies for dynamic vote reassignment. In *Proceedings of the 6th International Conference on Distributed Computing Systems*, pages 37–44, May 1986.

[4] Daniel Barbara, Hector Garcia-Molina, and Annemarie Spauster. Protocols for dynamic vote reassignment. In *Proceedings of the 5th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 195–205, ACM, 1986.

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

[6] Susan B. Davidson, Hector Garcia-Molina, and Dale Skeen. Consistency in partitioned networks. *ACM Computing Surveys*, 17(3):341–370, September 1985.

[7] 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.

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

[9] Hector Garcia-Molina and Daniel Barbara. Optimizing the reliabililty provided by voting mechanisms. In *Proceedings of the 4th International Conference on Distributed Computing Systems*, pages 340–346, October 1984.

[10] 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.

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

[12] 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, 1987.

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

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

[15] Michael Stonebraker. Concurrency control and consistency of multiple copies of data in distributed INGRES. *IEEE Transactions on Software Engineering*, SE-5(3):188–194, May 1979.

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

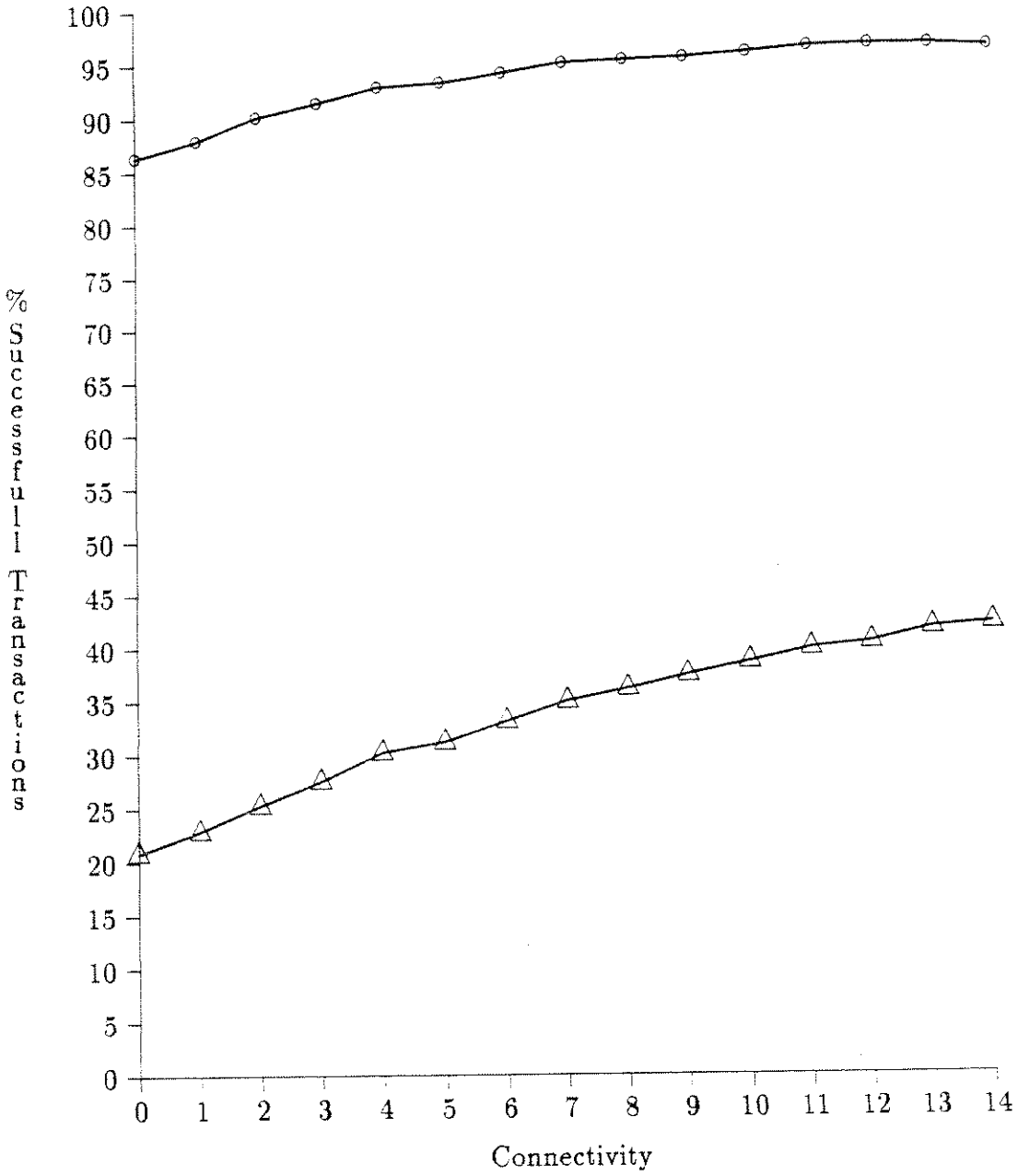

Figure 1: Metric Comparison, 7 sites, 50% reliability

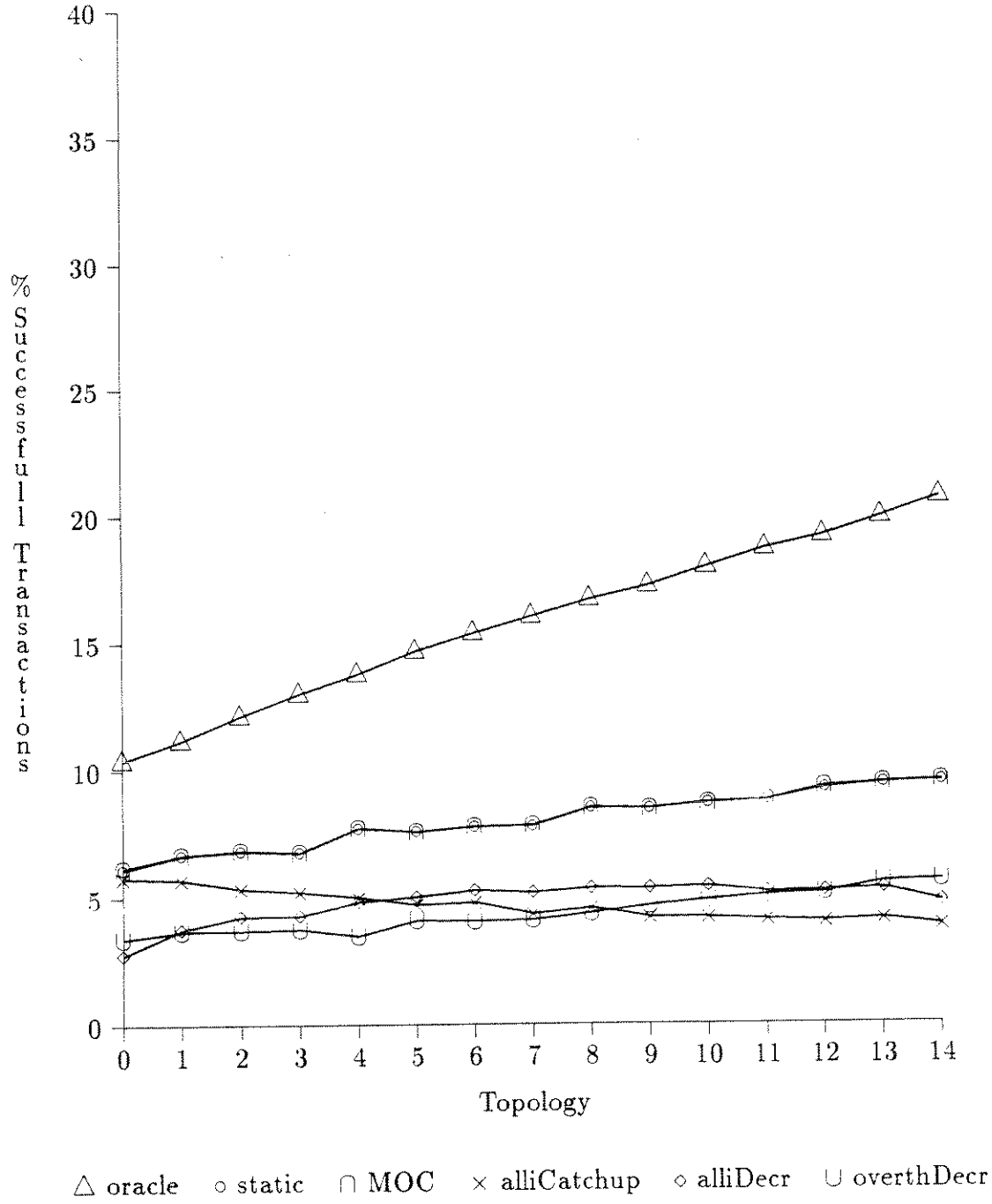

Figure 2: Protocol Comparison, 7 sites, 33% reliability

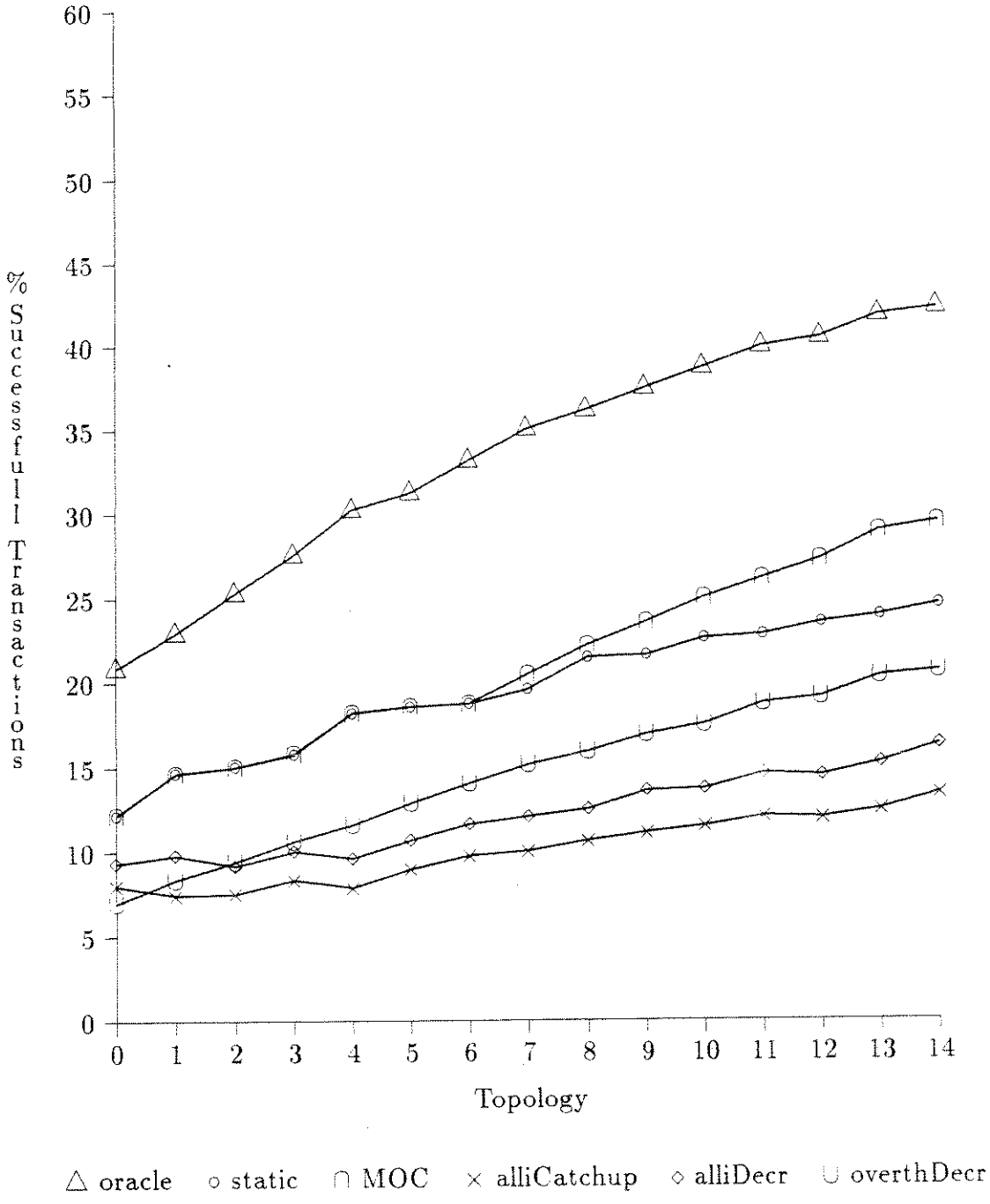

Figure 3: Protocol Comparison, 7 sites, 50% reliability

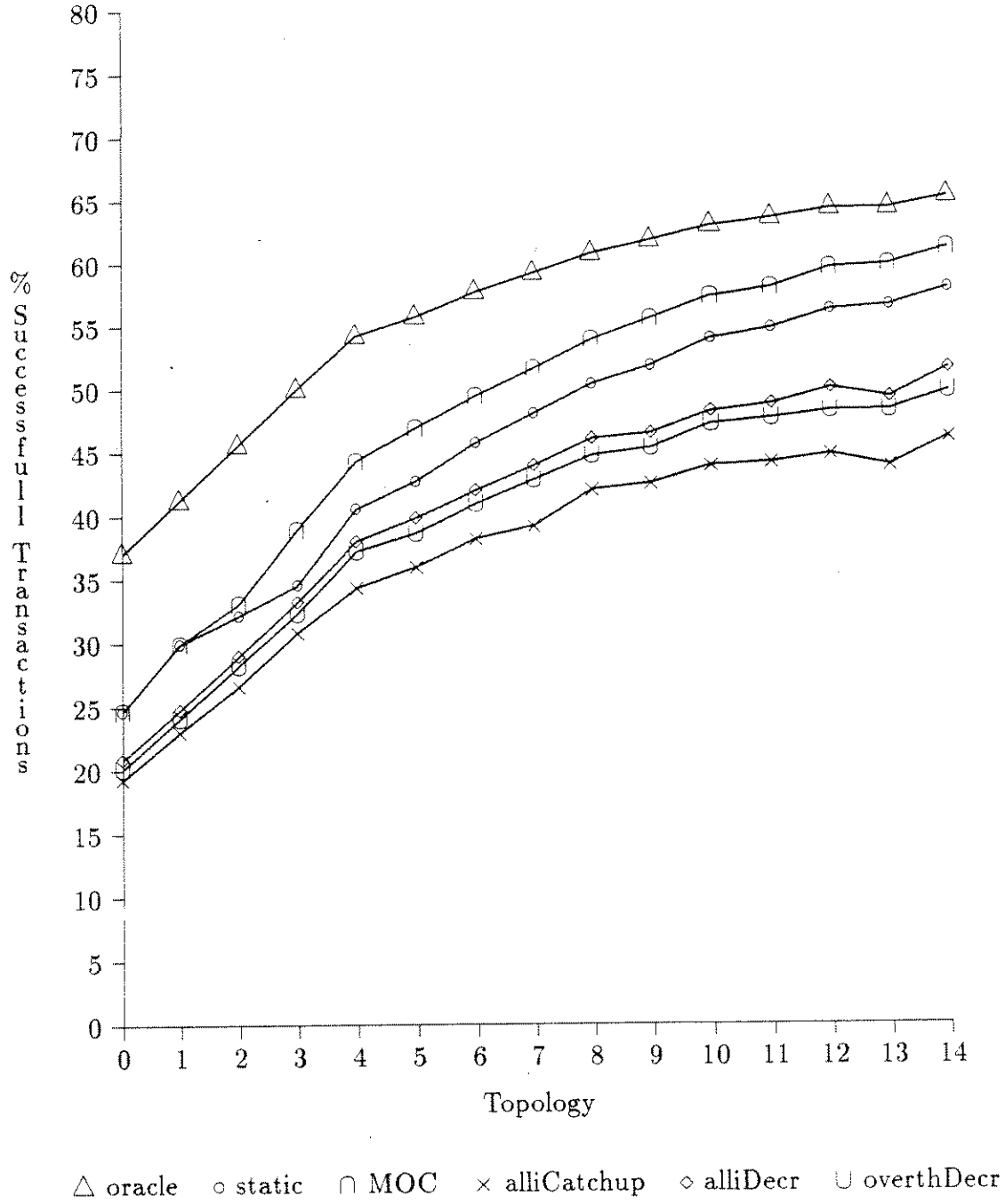

Figure 4: Protocol Comparison, 7 sites, 66% reliability

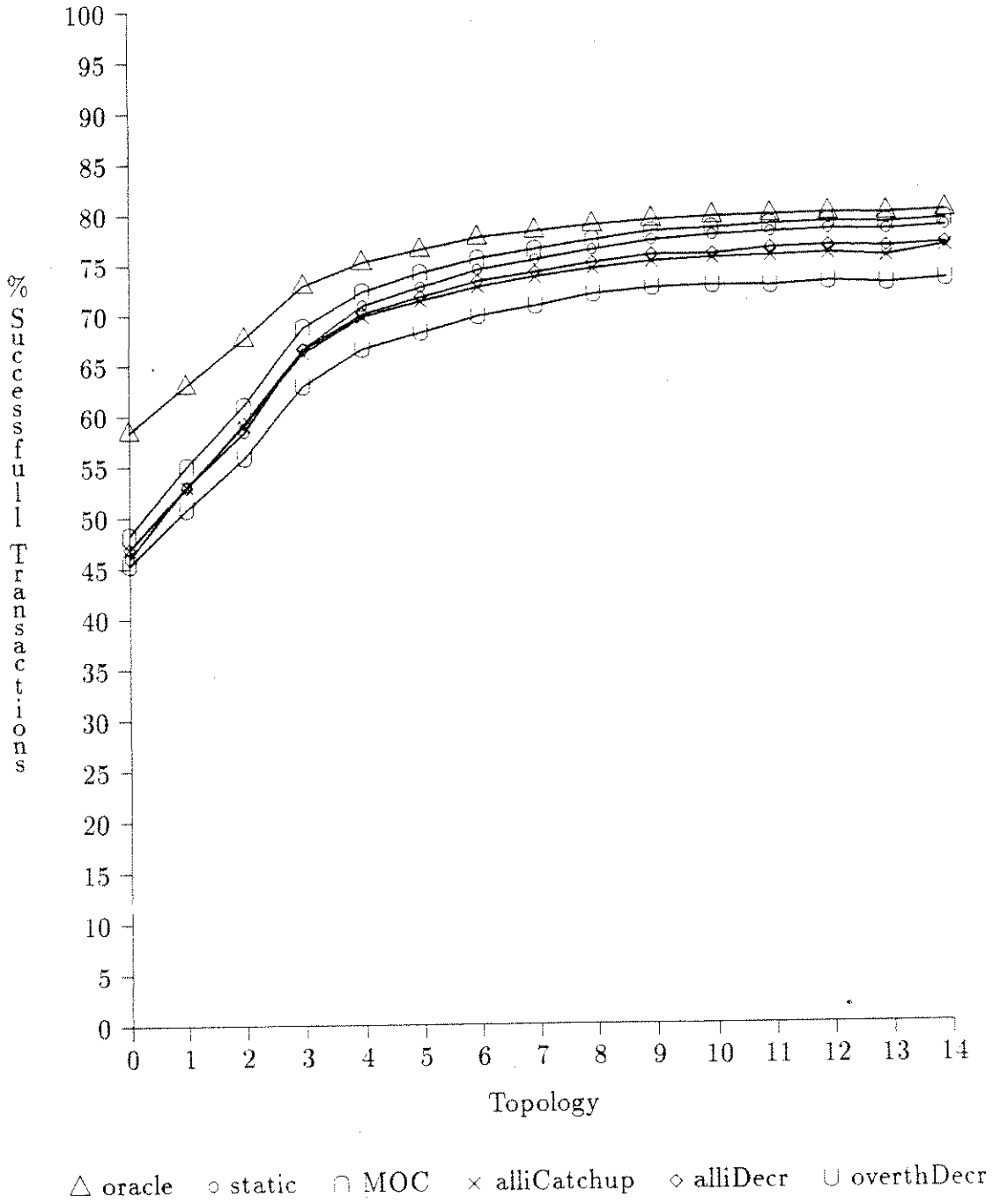

Figure 5: Protocol Comparison, 7 sites, 80% reliability

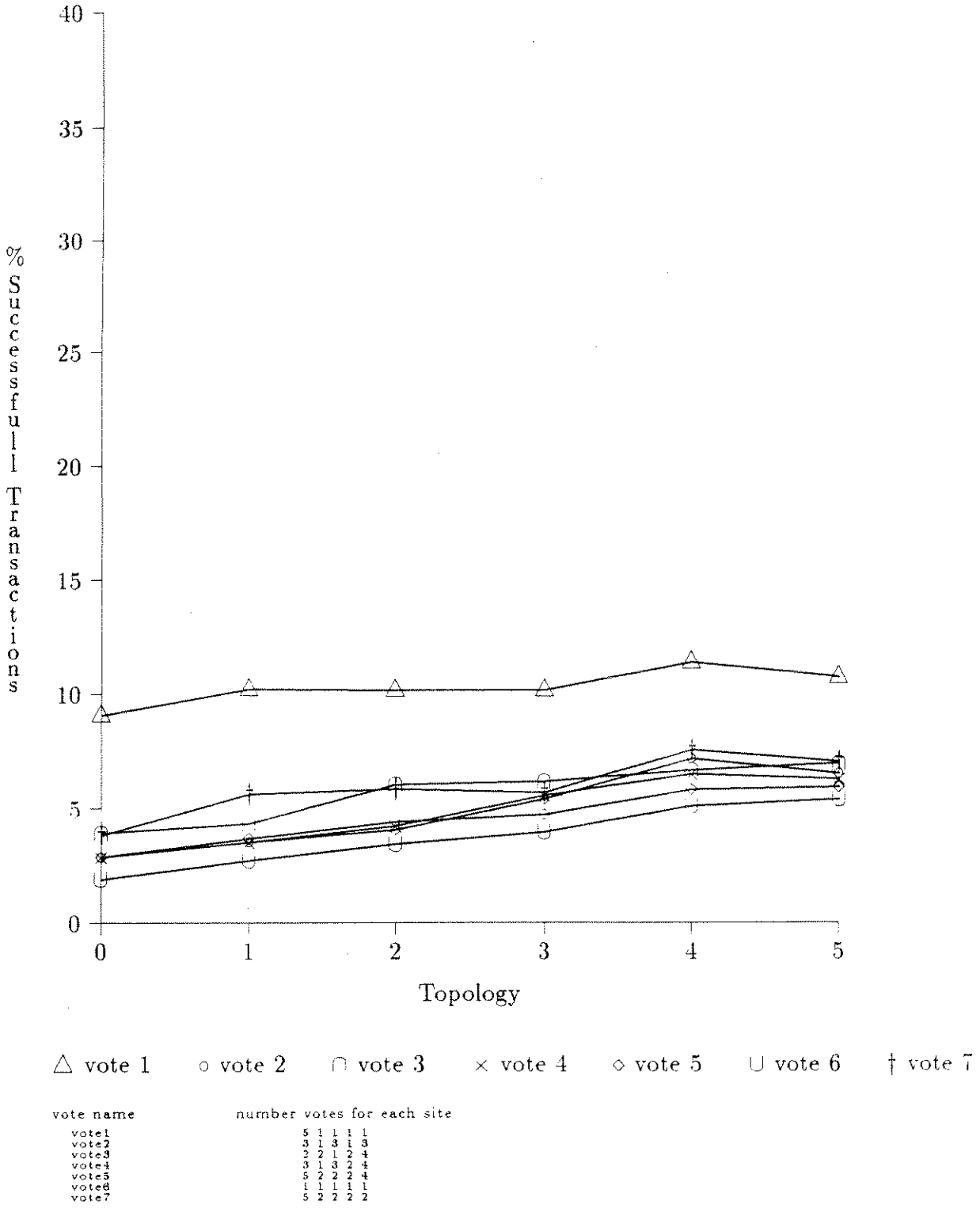

Figure 6: Static Protocol, 5 sites, 33% reliability

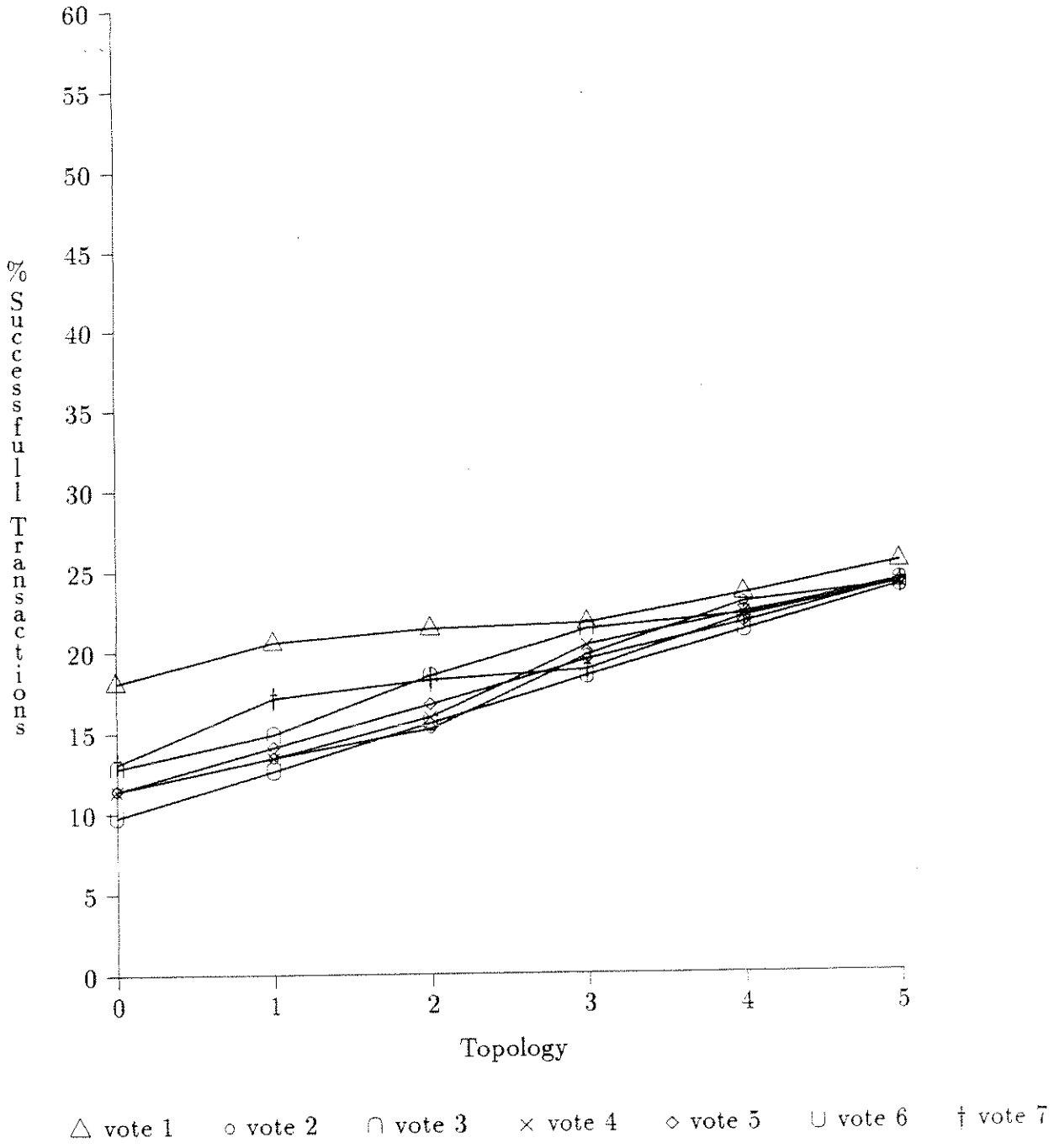

Figure 7: Static Protocol, 5 sites, 50% reliability

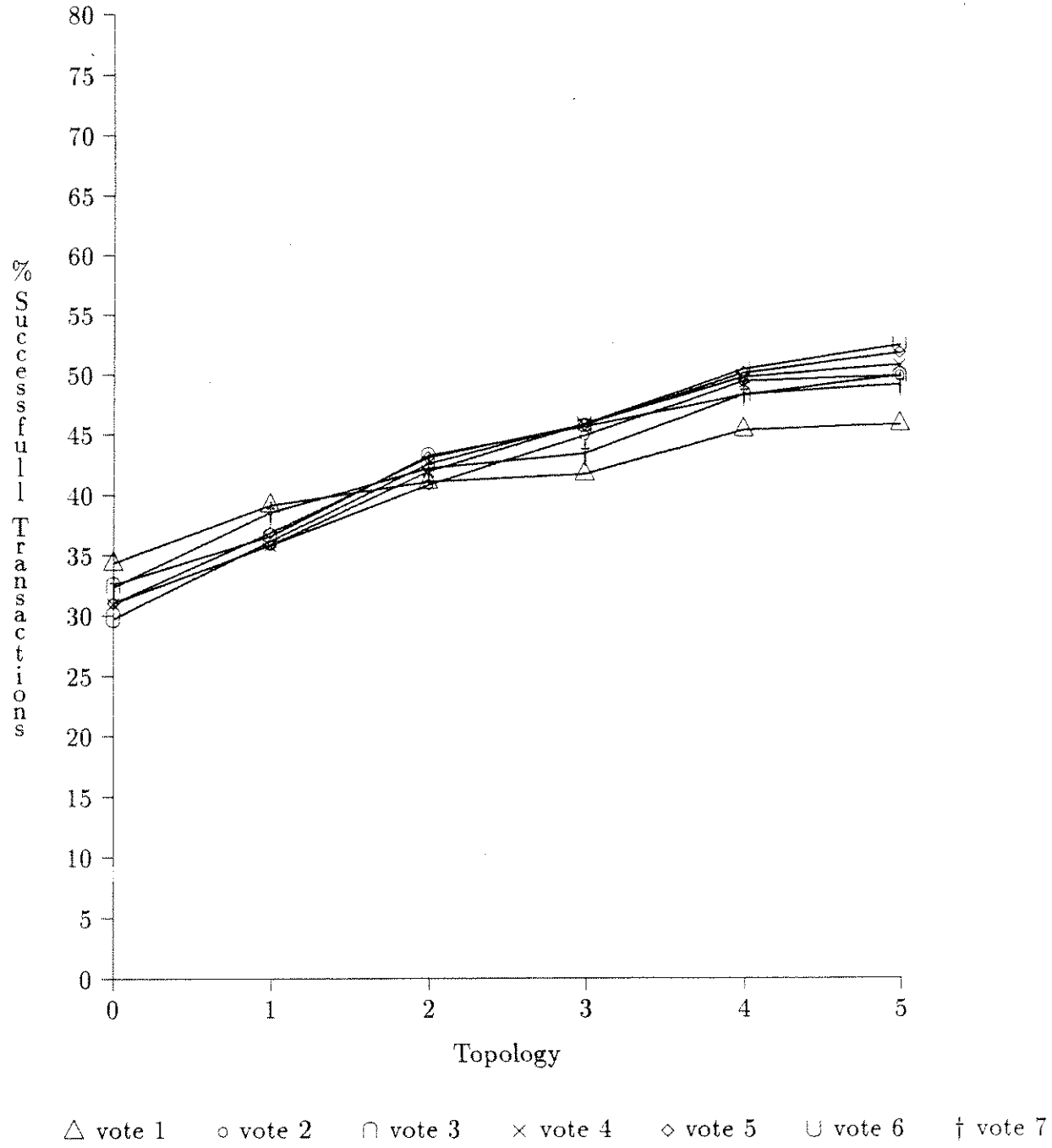

Figure 8: Static Protocol, 5 sites, 66% reliability

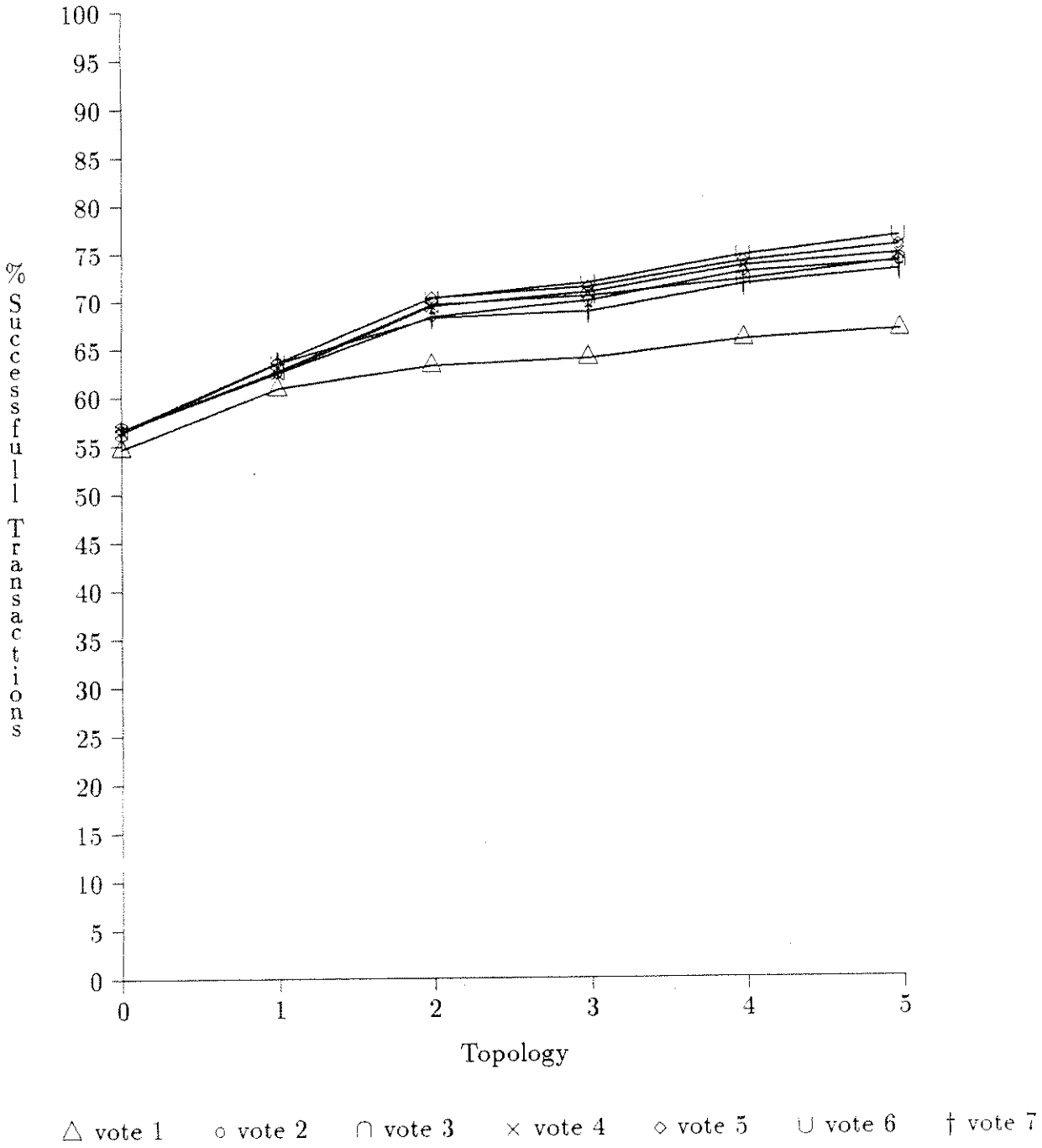

Figure 9: Static Protocol, 5 sites, 80% reliability

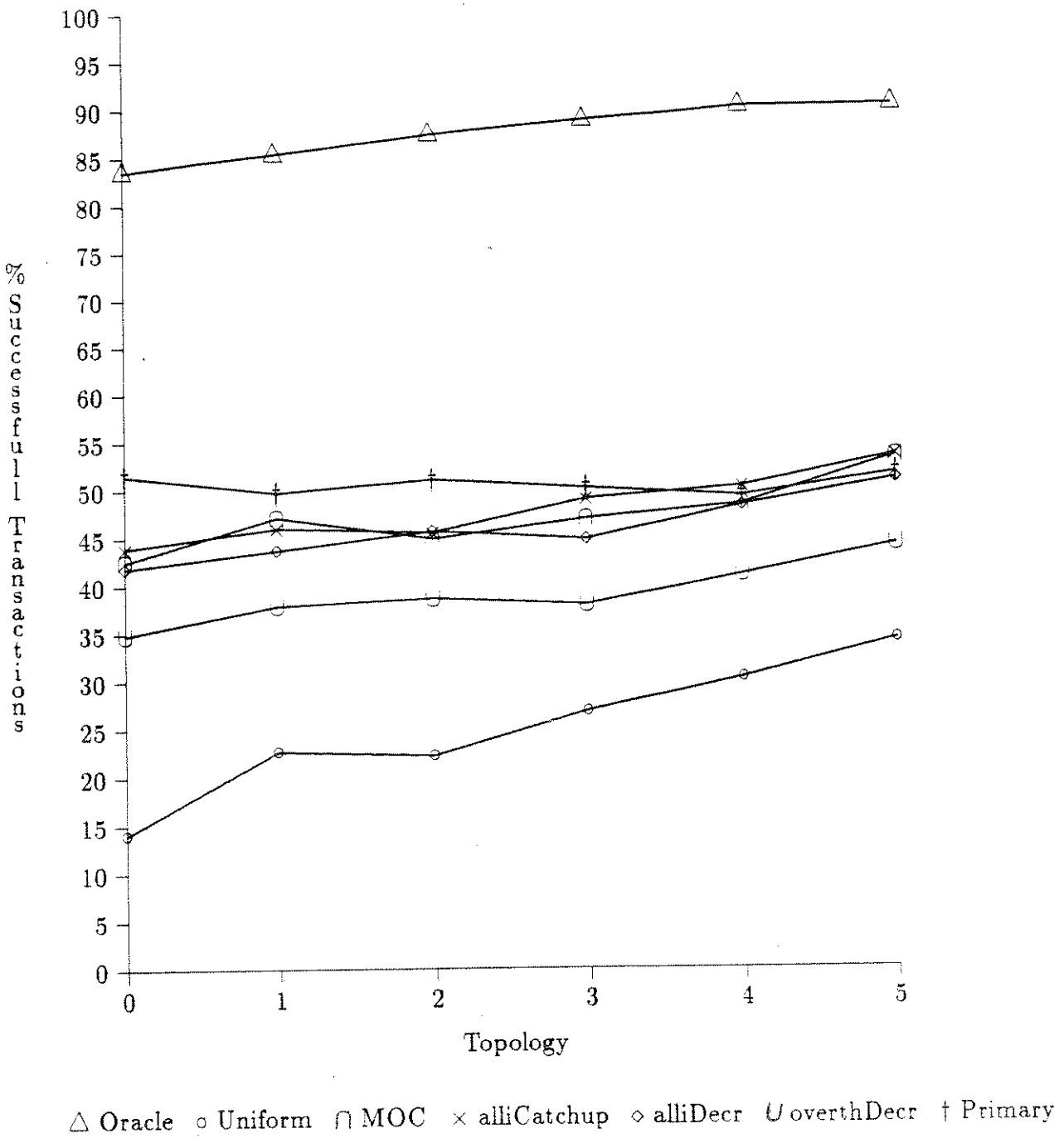

Figure 10: Protocol Comparison, OPT metric, 5 sites, 50% reliability