60

N91-23973

Measuring the Effects of Distributed Database Models On Transaction Availability Measures

Ravi Mukkamala

Department of Computer Science
Old Dominion University
Norfolk, Virginia 23529.
email: mukka@cs.odu.edu

Abstract

Data distribution, data replication, and system reliability are key factors in determining the availability measures for transactions in distributed database systems. In order to simplify the evaluation of these measures, database designers and researchers tend to make unrealistic assumptions about these factors. In this paper, we investigate the effect of such assumptions on the computational complexity and accuracy of such evaluations. We represent a database system with five parameters related to the above factors. Probabilistic analysis is employed to evaluate the availability of read-only and read-write transactions. We consider both the read-one/write-all and the majority-read/majority-write replication control policies. We conclude that transaction availability is more sensitive to variations in degrees of replication, less sensitive to data distribution, and insensitive to reliability variations in a heterogeneous system. The computational complexity of the evaluations is found to be mainly determined by the chosen distributed database model, while the accuracy of the results are not so much dependent on the models.

Keywords and phrases: Availability, Database models, Distributed Systems, Distributed Database Systems, Performance Evaluation, Probabilistic Analysis, Reliability, Transaction Availability

Measuring the Effects of Distributed Database Models
On Transaction Availability Measures

1 Introduction

A distributed database system is a collection of cooperating nodes each containing a set of data objects [1]. A user transaction can enter such a system at any of these nodes. The receiving node, some times referred to as the *coordinating* or *initiating* node, undertakes the task of locating the nodes that contain the data objects required by a transaction.

When we consider systems that require high guarantees for successful execution of transactions (especially for read-only transactions), it is important to consider transaction availability. Even though there are a number of availability (and reliability) metrics defined for computer systems [2,9], in this paper we choose two metrics: Start availability (TSA) and finish availability (TFA).

Transaction start availability (TSA) defines the probability with which a transaction can successfully start its execution. By our definition, a transaction is said to have a successful start when it can access all the required[2] copies of the data objects that it needs for its execution. For simplicity, we consider a data copy at a node to be *available* for access when that node is up and it is accessible from the node that is currently coordinating the execution of the transaction. A transaction can start its execution as soon as all the required data object copies are available.

Transaction finish availability (TFA) defines the probability with which a transaction can complete its execution, given that it has started its execution successfully. If execution times for transactions are negligible (as compared to the mean-time-to-fail of the components), then this reliability will be close to 1. However, since transactions take a finite but significant amount of time to execute, it is quite possible that the nodes that are involved in the execution of a transaction (and available at the start of execution) may

[1] In this paper, the basic unit of access in a database is referred to as a data object.

[2] The number of copies of an object that are required to be accessed by a transaction depends on the operation (read or write) and the replica copy control (e.g. read-one/write-all, majority) [3,18].

fail during its execution. In this case, the transaction is said to be aborted. In such cases, the execution needs to be restarted.

Formal definitions and evaluation of these two metrics (TSA and TFA) depend on several factors such as the fault model of the system (including the reliabilities of the system components), the transaction execution policy, the data distribution policy, the degree of data replication, the concurrency and commit protocols, and the characteristics of the given transaction [4,7,9]. In addition, TFA depends on the execution times of transactions.

Even though it is theoretically possible to formulate equations expressing the two metrics in terms of the above mentioned factors, the evaluation of these equations is extremely cumbersome and requires unreasonably high computation times. The evaluation of the exact values for these measures generally involves both analysis and simulation. Evaluation tools with such large execution times are certainly not acceptable to a database designer who needs to evaluate a number of such possible database configurations before arriving at a final design.

To overcome these problems, designers and researchers generally resort to approximation techniques [7,8,16]. These techniques reduce the computation time by making simplifying assumptions regarding data distribution, data replication, and transaction execution. The time complexity of these techniques primarily depends on the underlying model as well as the evaluation technique.

The effect of data distribution and replication models on evaluation of transaction response time has been measured in earlier studies [13]. These studies indicate that the *computational complexity* of a selected database model does not necessarily reflect the accuracy of the resulting performance evaluations. In fact, a model requiring computational time of [3] $O(n^2)$ has yielded results very close to those from a complex model with $O(n^n)$ complexity.

In this paper, we study the effect of data distribution, data replication, and fault models on the accuracy of transaction availability evaluations. We employ probabilistic analysis to arrive at the estimates for the desired values for six typical models.

The balance of this paper is outlined as follows. Section 2 formally

[3] Here, n denotes the number of nodes in a distributed system.

defines the problem under consideration. In Section 3, we describe a classification scheme for data distribution and replication policies. Section 4 illustrates the advantages of probabilistic analysis over simulation, and employs this technique to evaluate the measures for two different models. In Section 5, we compare the analysis methods for six models based on computational complexity, space complexity, and the accuracy of the measures. Finally, in Section 6, we summarize the obtained results, and suggest a general approach for design and analysis of these systems.

2 Problem Description

In this paper, a read-only transaction is characterized by the average number of data objects that it reads (i.e., its read-set size). Similarly, a read-write transaction is characterized by the number of data objects that it reads (read-set size), and the number of data objects that it updates (write-set size).

The problem of estimating the availability of a read-only transaction may be formulated as:

> Given the following parameters, estimate TSA_s and TFA_s for a read-only transaction that requires s data objects for read access.
>
> - n, the number of nodes in the database[4]
> - N, the index set for the nodes in the database; $N = \{1, 2, \ldots, n\}$
> - d, the number of data objects in the database
> - D, the index set for the data objects in the database; $D = \{1, 2, \ldots, d\}$
> - GD, the global data directory that contains the location of each of the d data objects; the GD matrix contains d rows and n columns, each of which is either a 0 or a 1; i.e., $GD_{ij} = 0$ or 1, $\forall i \in D$ and $\forall j \in N$
> - the reliability of the nodes in the network.

The problem of estimating the metrics for a read-write transaction can be similarly defined.

[4]Table 1 summarizes the notation used in this paper

Symbol	Description
a_i	The number of data objects accessed from the i^{th} group
c	The average number of copies of a data object
c_l	The number of copies of a data object in the l^{th} class
d	The number of data objects in the database
d_i	The number of data objects in the i^{th} class
g	The number of data object groups
k	Number of live nodes
n	Number of nodes
n_i	The number of nodes in the i^{th} class
p	The number of copy classes
q	The number of reliability classes
r	The average node reliability
r_i	The reliability of a node in the i^{th} class
s	The size of the read-set
A_1, A_2	Policies representing the data grouping
B_1, B_2	Policies representing limits on the data objects per node
C_1, C_2	Policies representing the degree of replication
D_1, D_2	Policies representing the copy distribution
E_1, E_2	Policies representing the component reliability
D	The index set for the data objects in the database
GA	Group access vector representing the number of objects accessed from each class or group
GD	Global data directory (or dictionary)
N	The index set for the nodes in the database
TSA_s	Transaction start availability of a read-only transaction with read-set size s (read-one/write-all policy)
$TSA'_{x,y}$	Transaction start availability of a read-write transaction with read-set size $x + y$ and write-set size y (read-one/write-all policy)
TSA''_s	Transaction start availability of a transaction with read-set size s (read-majority/write-majority policy)
x	The size of the read-only object set
y	The size of the read-write object set

Table 1: Notation

3 Model Description

As stated in the introduction, the primary objective of this paper is to investigate the effect of data distribution, replication, and fault models on availability estimations and the computational complexity of these evaluations.

To describe a data distribution, replication, and fault model, we characterize it with five orthogonal parameters:

A - Object grouping (or clustering)

B - Limits on the number of data objects per node

C - Degree of object replication (or the number of copies)

D - Constraints on distribution of object copies

E - Constraints on component reliability

We now discuss each of these parameters in detail.

Some distributed database systems allocate individual data objects [5, 10]. We categorize this strategy as A_1. In other systems, data objects are first partitioned into disjoint groups, and then the resulting groups are allocated [12,16,17]. Thus, the copies of all the data objects in a given group are allocated to the same set of nodes. We refer to this strategy as A_2.

Some database designers place no explicit limit on the number of data objects that may be placed at a node [7]. This strategy is named as B_1. Others restrict the number of data objects that may be placed at a given node. This may be attributed to storage limitations or for security reasons [11]. We refer to this strategy as B_2.

For simplicity, several analysis techniques assume that each data object has the same number of copies (or degree of replication) in the database system [6,16]. Some other techniques characterize the degree of replication of a database by the average degree of replication of data objects in that database [7]. In this paper, both these categories are referred to as C_1. Others treat the degree of replication of each data object independently. We refer to this as strategy C_2.

Some database designers and analysts assume that each data object (or group) copy is randomly distributed among the nodes in the distributed system [7]. We refer to this as D_1. Others assume some specific allocation schemes for data object (or group) copies [11]. Assuming complete knowledge of data copy distribution (GD) is one such assumption. Depending on the type of allocation, such assumptions may simplify the performance analysis [13]. This category is referred to as D_2.

Again for simplicity, some database designers and analysts assume that all components (nodes and links) in a distributed system have the same reliability factor [1]. In this paper, we only consider node failures and node repairs[5]. We let E_1 denote a policy where all nodes are assumed to have the same reliability characteristics, and E_2 denote a policy where nodes are classified based on their reliability characteristics.

Using this classification, any known data distribution, replication, and reliability policies may be categorized by these five parameters. For example, $< A_2, B_1, C_1, D_2, E_1 >$ represents a policy where

1. Data objects are *first grouped* and then allocated.
2. There is *no explicit limit* placed on the number of data objects (or groups) allocated to any node.
3. Each group has the same average degree of replication.
4. The copies of a group are distributed in some systematic manner among the nodes in the system.
5. All nodes in the system have identical reliability characteristics.

With these five parameters, we can describe thirty two basic policies. Several variations of these basic schemes are possible due to variations in systematic distributions (D_2), variations on the limits of data objects per node (B_2), and the types of grouping (A_2). Due to space limitations, in this paper we chose to present the results for six of these policies. Interested reader may refer to [14] for an analysis of other policies.

[5]That is, the underlying network structure almost always facilitates communication among live nodes.

We chose the following six policies to study the effect of the above mentioned parameters on availability computations:

Model 1: $< A_1, B_1, C_1, D_1, E_1 >$

Model 2: $< A_2, B_1, C_1, D_1, E_1 >$

Model 3: $< A_1, B_2, C_1, D_1, E_1 >$

Model 4: $< A_1, B_1, C_2, D_1, E_1 >$

Model 5: $< A_1, B_1, C_1, D_2, E_1 >$

Model 6: $< A_1, B_1, C_1, D_1, E_2 >$

Among these, Model 1 represents a simple system that is computationally attractive (as shown in Table 2). Model 2 reflects the effect of data grouping on the evaluation. Similarly, Model 3 reflects the effect of placing limits on number of data objects. Model 4 represents the effect of variations in number of copies of data objects on availability evaluation. Model 5 shows the effect of biased or *non-random* distributions of data objects on the evaluation. Finally, Model 6 reflects the effect of non-homogeneous environment (i.e., different node reliability characteristics) on transaction availability evaluation.

In the following section, we derive closed-form expressions for the average transaction availabilities for Models 1 and 2.

4 Probabilistic Computation of the Availabilities

There are several approaches for computing the availability of a given transaction in a database. These computations assume a given data distribution, data replication, and fault models. We now look at two such methods: simulation and probabilistic analysis.

Using simulation, one can generate the data distribution matrix (GD) based on the data distribution and replication model. One can also generate the reliabilities for each of the nodes in the system [6]. Similarly, one can generate all possible transactions (with different read-sets and write-sets) that

[6] Here, we ignore the possibility of network partitioning, and thereby ignore link reliability factor.

can be received at each of the nodes in the network. For each such transaction received by the system, the data distribution matrix can be searched, and its ability to access all the required data objects may be verified. In addition to generating transactions, we should also generate node failures and node repairs in the time domain. Thus, some transactions may not be successful due to the inaccessibility of one or more data objects that they require (due to node failures). With such statistics (of successful/unsuccessful transactions) in hand, we can obtain the average availability of a transaction of a given size. This average corresponds to a single distribution matrix. The generation and evaluation process may have to be repeated sufficient times to get the required confidence in the final result. Since there are d data objects, there are $\binom{d}{s}$ possible transactions with read-set [7] size s, and there are n nodes where each of these may be received. Given a *transaction*, and the *node* where it is received, determining the state (successful/unsuccessful) of a transaction takes at least $O(nd)$ computations (i.e., to scan the columns of the GD matrix corresponding to available nodes). If the distribution matrix is generated k times, then the evaluation of the desired average set size for a transaction of size s takes $O(kn^2d\binom{d}{s})$ time. In general, k is a function of the number of copies, the number of data objects, the number of nodes, and the data distribution model, and it could be very high. Suppose $d = 100$, $s = 10$, and $n = 10$, then this method requires approximately $10^{17}k$ computations. Even for reasonable values of k, this is an unreasonably high computation time.

To avoid this large evaluation time, we adopt probabilistic analysis. In this analysis, we essentially study the given data distribution and reliability model and arrive at an expression for the average transaction availability for a given read-set (or write-set) size. With probabilistic analysis, some data distribution models (e.g., Models 1 and 3) may require insignificant amounts of computation. Some may need moderate computation times (e.g., Models 2 and 6), whereas others may need large computation times (e.g., Models 4 and 5). Regardless of the model, all these need considerably less computation time (with more accuracy of results) than the corresponding simulation methods.

We now illustrate the probabilistic method of analysis by applying it for

[7] The corresponding term for write-sets of update transactions may be easily written.

Models 1 and 2. Expressions for other models may be derived in a similar manner. Interested reader may find the details of these derivations in [14].

4.1 Derivation of Reliability Metrics for Model 1

Model 1, designated as $< A_1, B_1, C_1, D_1, E_1 >$ assumes the following about the data distribution and replication:

[R1] The data objects are allocated individually (i.e. not grouped) to the nodes.

[R2] There are no limits placed on the number of data objects that may be placed at each node.

[R3] The average degree of replication (c) of a data object is given.

[R4] The copies of a data object are allocated randomly.

[R5] Each node in the system has identical reliability ($= r$).

Further, to simplify the illustration of the current analysis, we make the following assumptions regarding the distribution of groups, and the participating node set determination:

[R6] Each transaction is equally likely to access any data object.

[R7] The transactions that enter the distributed system are coordinated by a set of *reliable servers* that search the distributed database system (i.e., the availability of nodes and their dictionaries) for the availability of the required data objects.

Due to Rule R7, we will not distinguish transactions that are received at different locations in the system. Thus, we will disregard the originating node as a parameter in this analysis[8].

[8]The analysis can easily be extended to a situation where transactions received at an unavailable node are automatically considered as unsuccessful.

4.1.1 Derivation of Availability for Read-only Transactions

Let us consider a read-only transaction T_1 with s objects in its read-set and received at one of the servers. Let us also assume that the copy control algorithm follows a read-one/write-all policy. Thus T_1 needs to access *any one* of the c copies of a data object that it requires.

Given that exactly k of the n nodes are available (i.e., up), the probability that at least one copy of a given data object is available is given by:

$$P_{k,1} = 1 - \frac{\binom{n-k}{c}}{\binom{n}{c}} \tag{1}$$

By definition of the read-one/write-all policy, $P_{k,1}$ represents the probability that a data object is available for read access in the system. Since each data object is allocated independently to the nodes in the system (by Rules R1 and R2), the probability that all s data objects required by T_1 are available for read access within these k nodes can then be expressed as:

$$P_{k,s} = P_{k,1}^s = \left[1 - \frac{\binom{n-k}{c}}{\binom{n}{c}}\right]^s \tag{2}$$

Assuming the reliability of any given node to be r (from Rule R5), the probability that T_1 has successfully started is:

$$\begin{aligned} TSA_s &= \sum_{k=1}^{n} \binom{n}{k} r^k (1-r)^{n-k} P_{k,s} \\ &= \sum_{k=1}^{n} \binom{n}{k} r^k (1-r)^{n-k} \left[1 - \frac{\binom{n-k}{c}}{\binom{n}{c}}\right]^s \end{aligned} \tag{3}$$

Given that T_1 has successfully started, we will now compute the probability with which it can be successfully completed. Let us assume that n_s nodes are involved in the execution of T_1, and that it has an execution time of t units. Now, in order for T_1 to be successful, all these n_s nodes have to be available for at least t units of time, given that they were available at the start of execution. Assuming an exponential distribution for time between node failures with a failure rate of λ, the probability that a node which is available at time zero is available throughout time t is given by:

$$A_t = e^{-t\lambda} \tag{4}$$

From here, the probability that none of the n_s nodes have failed during time t is given by:

$$\begin{aligned} TFA_s &= A_t^{n_s} \\ &= e^{-n_s t\lambda} \end{aligned} \tag{5}$$

Estimating n_s for transaction T_1 is a complex problem. This problem has been well investigated and the details of the solutions may be found in [15]. In this paper, we assume that n_s for T_1 has been obtained *a priori* for a given data distribution and fault model.

4.1.2 Derivation of Availability for Read-write Transactions

Let us now consider a read-write transaction T_2 with s objects in its read-set and y objects in its write-set. Let us assume that for a given read-write transaction write-set$\subseteq$read-set [3,7]. Thus, among the s data objects, y objects are both read and written, while $x = s - y$ data objects are only read. (Note that the intersection of the read-only and the read-write sets of the data objects is empty.) Since the replication control algorithm follows a read-one/write-all policy, T_2 needs to access all c copies of the y data objects and any one copy of the x data objects.

Given that exactly k of the n nodes are available (i.e., up), the probability that *all* c copies of a given data object are available is given by:

$$P'_{k,1} = \frac{\binom{k}{c}}{\binom{n}{c}} \tag{6}$$

Since each data object is allocated independently to the nodes in the system (by Rules R1 and R2), the probability that all y data objects required by T_2 are accessible for update is expressed as:

$$P'_{k,y} = \left[\frac{\binom{k}{c}}{\binom{n}{c}}\right]^y \tag{7}$$

Similarly, the probability that all x data objects are available for read access may be computed as:

$$P_{k,x} = \left[1 - \frac{\binom{n-k}{c}}{\binom{n}{c}}\right]^x \tag{8}$$

From here, the probability that T_2 is successfully started may be computed as:

$$\begin{aligned} TSA'_{x,y} &= \sum_{k=1}^{n} \binom{n}{k} r^k (1-r)^{n-k} P'^{y}_{k,y} P^{x}_{k,x} \\ &= \sum_{k=1}^{n} \binom{n}{k} r^k (1-r)^{n-k} \left[\frac{\binom{k}{c}}{\binom{n}{c}}\right]^y \left[1 - \frac{\binom{n-k}{c}}{\binom{n}{c}}\right]^x \end{aligned} \tag{9}$$

The finish availabilities for T_2 may be similarly computed using Equations (4) and (5) where n_s is now replaced by $n_{x,y}$ [14].

4.1.3 Derivation of Availability for Transactions with Majority Consensus

In the above two sections, we dealt with read-one/write-all replication control policy. The majority consensus protocols [18] which require the accessibility of at least a majority of the total copies of a data object for both read and write operations are very attractive in a failure prone environment. Since both read and write operations require the same number of copies of a data object, in this analysis we do not distinguish between read-only and update transactions. Here, we simply refer to T_1 as a transaction.

Let $m = \lceil \frac{c+1}{2} \rceil$ represent the majority of copies. Then the expression for start availability for T_1 is given as:

$$TSA''_s = \sum_{k=m}^{n} \binom{n}{k} r^k (1-r)^{n-k} \left[\sum_{l=m}^{c} \frac{\binom{k}{l}\binom{n-k}{c-l}}{\binom{n}{c}}\right]^s \tag{10}$$

Similarly, the expression for the finish availability for T_1 may be expressed as:

$$\begin{aligned} TFA_s &= A_t^{n_s} \\ &= e^{-n_s t\lambda} \end{aligned} \tag{11}$$

where n_s now represents the average number of nodes accessed for executing T_1 with the majority consensus protocol [15].

4.2 Derivation of Transaction Availability for Model 2

Model 2, designated as $< A_2, B_1, C_1, D_1, E_1 >$ is similar to Model 1, except that the data objects are now grouped, and the groups are then allocated to nodes in the system. This may be described as:

[R9] The data objects are first grouped and the groups are then allocated, to the nodes. Let the d data objects be partitioned into t distinct groups. Let d_k represent the number of data objects in group k. Thus, $\sum_{i=1}^{t} d_i = d$.

[R10] There are no limits placed on the number of groups that may be placed at each node.

[R11] The degree of replication is the same for each group (c).

[R12] The copies of a group are allocated randomly.

[R13] Each node in the system has identical reliability (r).

Again, to simplify analysis, we make the following assumptions:

[R14] Each transaction is equally likely to access any data object.

[R15] The transactions that enter the distributed system are coordinated by a set of *reliable servers* that search the distributed database system (i.e., the availability of nodes and their dictionaries) for the availability of required data objects.

4.2.1 Derivation of Availability for Read-only Transactions

Once again let us consider transaction T_1 executing under a read-one/write-all policy. Given that k of the n nodes are available (i.e., up), the probability that at least one copy of *group* k is available is given by:

$$1 - \frac{\binom{n-k}{c}}{\binom{n}{c}} \tag{12}$$

If the vector $GA =< a_1, a_2, \ldots, a_t >$ represents the number of data objects accessed by T_1 from each of the t groups, then the probability that T_1 is

successfully started may be computed as:

$$TSA_s = \sum_{GA} Pr(GA) \sum_{l=1}^{n} \binom{n}{l} r^l (1-r)^{n-l} \prod_{k=1}^{t} \left[1 - \frac{\binom{n-l}{c}}{\binom{n}{c}} \right]^{f(k)} \quad (13)$$

$$Pr(GA) = \frac{\binom{d_1}{a_1}\binom{d_2}{a_2}\cdots\binom{d_t}{a_t}}{\binom{d}{s}} \quad (14)$$

$$f(k) = \begin{cases} 1 & \text{if } a_k > 0 \\ 0 & \text{otherwise} \end{cases} \quad (15)$$

$$GA = < a_1, a_2, \ldots, a_t >, \quad \sum_{k=1}^{t} a_k = s \text{ and } \forall k\ 1 \le k \le t\ 0 \le a_k \le d_k \quad (16)$$

When data objects are equally distributed among the groups (i.e., $d_1 = d_2 = \ldots = d_t = \frac{d}{t}$), then this expression may be further simplified as:

$$TSA_s = \sum_{l=1}^{n} \sum_{k=1}^{t} \binom{n}{l} r^l (1-r)^{n-l} \binom{t}{k} \left[1 - \frac{\binom{n-l}{c}}{\binom{n}{c}} \right]^k \left[\frac{\binom{n-l}{c}}{\binom{n}{c}} \right]^{t-k} \frac{\binom{\frac{dk}{t}}{s}}{\binom{d}{s}} \quad (17)$$

The expression for TFA_s is the same as in Equation (5).

4.2.2 Derivation of Availability for Read-write Update Transactions

Let us consider transaction T_2 which requires x objects for read-only operations and y data objects for read and write operations ($s = x + y$). Thus we need to define two GA vectors for read-only and read-write data object sets:

$$GA' = < a'_1, a'_2, \ldots, a'_t >, \quad \sum_{k=1}^{t} a'_k = x \text{ and } \forall k\ 1 \le k \le t\ 0 \le a' \le d_k$$

$$GA'' = < a''_1, a''_2, \ldots, a''_t >, \quad \sum_{k=1}^{t} a''_k = y \text{ and } \forall k\ 1 \le k \le t\ 0 \le a'' \le d_k - a'_k$$

In computing $TSA'_{x,y}$ we should recall that if a data object is write accessible under a given node availability conditions, it is also read accessible. However the reverse is not true. These two facts are made use of in deriving the following expression for $TSA'_{x,y}$:

$$
\begin{aligned}
TSA'_{x,y} &= \sum_{GA'} \sum_{GA''} Pr(GA') Pr(GA'') \sum_{l=1}^{n} \binom{n}{l} r^l (1-r)^{n-l} \\
&\quad \prod_{k=1}^{t} \left[1 - \frac{\binom{n-l}{c}}{\binom{n}{c}} \right]^{f'(k)} \prod_{k=1}^{t} \left[\frac{\binom{l}{c}}{\binom{n}{c}} \right]^{f''(k)} \qquad (18) \\
Pr(GA') &= \frac{\binom{d_1}{a'_1}\binom{d_2}{a'_2} \cdots \binom{d_t}{a'_t}}{\binom{d}{x}} \\
Pr(GA'') &= \frac{\binom{d_1 - a'_1}{a''_1}\binom{d_2 - a'_2}{a''_2} \cdots \binom{d_t - a'_t}{a''_t}}{\binom{d-x}{y}} \\
f'(k) &= \begin{cases} 1 & \text{if } a''_k = 0 \wedge a'_k > 0 \\ 0 & \text{otherwise} \end{cases} \\
f''(k) &= \begin{cases} 1 & \text{if } a''_k > 0 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

As before, when data objects are equally distributed among the groups (i.e. $d_1 = d_2 = \ldots = d_t = \frac{d}{t}$), this expression may be simplified as:

$$
\begin{aligned}
TSA'_{x,y} &= \sum_{l=1}^{n} \sum_{k_1=1}^{t} \sum_{k_2=0}^{t-k_1} \binom{n}{l} r^l (1-r)^{n-l} \binom{t}{k_1} \binom{t-k_1}{k_2} \left[\frac{\binom{l}{c}}{\binom{n}{c}} \right]^{k_1} \\
&\quad \left[1 - \frac{\binom{n-l}{c} + \binom{l}{c}}{\binom{n}{c}} \right]^{k_2} \left[\frac{\binom{n-l}{c}}{\binom{n}{c}} \right]^{t-k_1-k_2} \frac{\binom{\frac{dk_1}{t}}{y}}{\binom{d}{y}} \frac{\binom{\frac{d(k_1+k_2)}{t} - y}{x}}{\binom{d-y}{x}} \qquad (19)
\end{aligned}
$$

The finish availability $TFA_{x,y}$ may be computed using Equation (5) where n_s is now replaced by $n_{x,y}$ which is assumed to be known a priori in this paper.

4.2.3 Derivation of Availability for Transactions with Majority Consensus

As described in Section 4.1.3, under the majority consensus protocol both the read-set and read-write set are treated in the same way for access probability computations. Thus, we only consider a read-only transaction with a read-set size of s. The expression for TSA''_s can now be written as:

$$TSA''_s = \sum_{GA} Pr(GA) \sum_{l=1}^{n} \binom{n}{l} r^l (1-r)^{n-l} \prod_{k=1}^{t} \left[\sum_{l'=m}^{c} \frac{\binom{l}{l'}\binom{n-l}{c-l'}}{\binom{n}{c}} \right]^{f(k)} \quad (20)$$

$$m = \left\lceil \frac{c+1}{2} \right\rceil \quad (21)$$

where $Pr(GA)$ and $f(k)$ are as defined in Equations (14) and (15).

Once again, when data objects are equally distributed among the groups (i.e. $d_1 = d_2 = \ldots = d_t = \frac{d}{t}$), this expression may be written as:

$$TSA''_s = \sum_{l=m}^{n} \sum_{k=1}^{t} \binom{n}{l} r^l (1-r)^{n-l} \binom{t}{k} \left[\sum_{l_1=m}^{\min(l,c)} \frac{\binom{l}{l_1}\binom{n-l}{c-l_1}}{\binom{n}{c}} \right]^{k} \left[1 - \sum_{l_1=m}^{\min(l,c)} \frac{\binom{l}{l_1}\binom{n-l}{c-l_1}}{\binom{n}{c}} \right]^{t-k} \frac{\binom{\frac{dk}{t}}{s}}{\binom{d}{s}} \quad (22)$$

5 Comparison of the Availabilities for the Six Models

As mentioned in the introduction, the main objective of this paper is to determine the effect of data distribution, replication, and fault models on the estimation of transaction availability. To achieve this, we evaluate the desired measure using six different models. The comparison of these evaluations is based on computational time, storage requirement, and the average values obtained.

Due to space limitations, we cannot present the detailed derivations for the average values for Models 3-6. The final expressions, however, are summarized in the appendix.

5.1 Computational Complexity

We now analyze each of the evaluation methods (for Models 1-6) for their computational complexity.

- Let us refer to Model 1. From Equations (3) and (9), it is clear that computation of TSA_s and $TSA_{x,y}$ take $O(cn^2)$ time [9]. Similarly, from Equation (10), it is clear that the computation of TSA''_s requires $O(c^2n^2)$ time.

- We now derive this complexity term for Model 2. Let us first look at the computation of TSA_s. From Equation (14), we derive that the computation of $Pr(GA)$ requires $O(s)$ time. The number of GAs generated is approximately $O(s^t)$ where t represents the number of data object groups. Given a GA vector and $Pr(GA)$, computation of TSA_s requires $O(nct+n^2)$ arithmetic operations (from Equation (18)). Thus the evaluation of TSA_s requires $O(s^t\,(nct + n^2 + s))$ time. Similarly, we can conclude that $TSA'_{x,y}$ requires $O(x^t y^t(nct + n^2 + s))$ time (Equation (19)), and TSA''_s requires $O(s^t(nc^2t + n^2 + s))$ time (Equation (20)).

- For Model 3, the computational complexity for TSA_s is $O(n^2+n(s+c))$ (Equation (23)). Similarly, $TSA'_{x,y}$ and TSA''_s require $O(n^2+n(c+s))$ and $O(n^2 + n(c^2 + s))$ respectively (Equations (24) and (25)).

- The computational complexity for Model 4 depends on the number of copy categories. Assuming that $s < d_k$ for $k = 1, 2, \ldots p$, we can generate approximately s^p different CA vectors. Thus the computation of TSA_s requires $O(s^p(n^2+npc+s))$ time. To compute TSA', we need to compute the number of possible CA' and CA'' vectors. There are approximately x^p CA' vectors and y^p CA'' vectors. Thus, $TSA'_{x,y}$ requires $O(x^p y^p(npc + n^2 + s))$ time. Similarly, we can conclude that TSA''_s requires $O(s^p(npc^2 + n^2 + s))$.

- In Model 5, we assume that the entire data dictionary information is available to us. Given a GD matrix and a node status vector S,

[9] Here, we are assuming that the evaluation of the terms $\binom{p}{q}$ and p^q takes $O(q)$ and $O(1)$ time respectively.

computation of $f(S)$, $f'(S)$, and $f''(S)$ require $O(nd)$ time to search the matrix. Given n, there are 2^n possible S vectors. Thus the computations of TSA, TSA', and TSA'' require $O(2^n(nd+s))$ time.

- In Model 6, the number of NA vectors generated is $(n_1+1)(n_2+1)\ldots(n_q+1)$. For simlification, we approximate it as $\left(\frac{n}{q}+1\right)^q$. Given a NA vector, the computation of TSA, TSA', and TSA'' require $O(s+c+q)$, $O(s+c+q)$ and $O(sc+c^2+qc)$ time respectively. Thus the three metric evaluations require $O((\frac{n}{q}+1)^q(s+c+q))$, $O((\frac{n}{q}+1)^q(s+c+q))$, and $O((\frac{n}{q}+1)^q(cs+c^2+cq))$ time respectively.

These complexities are summarized in Table 2. From this table it may be observed that models 1 and 2 are computationally very attractive. The complexity of evaluations with models 2,4, and 6 depend on the number of groups, the number of copy variations, and the number of reliability variations respectively. For systems with a large number of nodes, evaluations with model 5 are very expensive.

5.2 Space Complexity

We now discuss the space complexity for the six models:

- Models 1 and 3 just require the values of d, c, s, r and n. Thus the storage requirement is $O(1)$
- Since Model 2 requires that the d_i values be stored, and that the GA vectors be generated, it requires $O(t)$ storage, where t is the number of data groups.
- Model 4 requires $O(p)$ storage to contain the p copy classes.
- Model 5 requires $O(nd)$ storage for the GD matrix.
- Model 6 requires $O(q)$ storage to contain the node reliability class information.

Thus, Model 5 has the largest storage requirement. These complexities are summarized in Table 3.

Model	Computational Complexity		
	Read-only	Read-write	Majority
1	$O(cn^2)$	$O(cn^2)$	$O(c^2n^2)$
2	$O(s^t(nct + n^2 + s))$	$O(x^ty^t(nct + n^2 + s))$	$O(s^t(nc^2t + n^2 + s))$
3	$O(n^2 + nc + ns)$	$O(n^2 + nc + ns)$	$O(n^2 + nc^2 + ns)$
4	$O(s^p(npc + n^2 + s))$	$O(x^py^p(npc + n^2 + s))$	$O(s^p(npc^2 + n^2 + s))$
5	$O(2^n(nd + s))$	$O(2^n(nd + s))$	$O(2^n(nd + s))$
6	$O((\frac{n}{q} + 1)^q(s + c + q))$	$O((\frac{n}{q} + 1)^q(s + c + q))$	$O((\frac{n}{q} + 1)^q(cs + c^2 + cq))$

Table 2: Computational Complexities for the Evaluation of Availabilities

Model	Space Complexity
1	$O(1)$
2	$O(t)$
3	$O(1)$
4	$O(p)$
5	$O(nd)$
6	$O(q)$

Table 3: Space Complexities for the Evaluation of Availabilities

5.3 Comparison of the Availabilities

In order to compare the effectiveness of each of these models, we have evaluated availabilities for a wide range of parameters. Due to space limitations, in this paper, we only present a small subset of these results. Similarly, since TFA_s, $TFA'_{x,y}$, and TFA''_s are found to be insensitive to variations in models, we are not presenting these results here. We only present the results for the transaction start availabilities. These results are summarized in Figures 1-7.

Figures 1-3 compare the availabilities obtained from the six models. The following assumptions are made for models 1-6:

1. In Model 2, we assume that the d data objects are grouped into n data groups each containing d/n data objects. This is similar to the assumptions in [13].

2. In Model 3, we assume that each of the n nodes in the system is allocated *exactly* the same number of data objects (equal to dc/n).

3. In Model 4, we assume that $d/2$ data objects have c copies, $d/4$ data objects have $c+1$ copies, and the rest have $c-1$ copies. This keeps the average copies the same (i.e., c) but brings a copy variation factor into consideration.

4. In Model 5, we assume that the d data objects are allocated systematically so that the copies of the i^{th} data object are allocated, in a circular manner, to the nodes starting from $(i \oplus n) + 1$.

5. In Model 6, we assume that $n/3$ nodes have reliability $r - 0.1$, $n/3$ have reliability $r + 0.1$ and the rest have a reliability r. [10]

Figure 1 summarizes the results for read-only transactions with read-one/write-all policy. Figure 2 presents these results for transactions (read-only or read-write) with majority-read/majority-write protocol. Finally, Figure 3 summarizes the results for read-write transactions with read-one/write-all policy. From these results, we make the following observations:

[10] When $r = 0.95$, we assume that $n/3$ nodes have reliability $r - 0.5$, $n/3$ have reliability $r + 0.05$ and the rest have a reliability r.

- For read-only transactions (with read-one/write-all policy),

  (i) Evaluations with models 1 and 3 are close over the entire range of s and r.

  (ii) Evaluations with models 2 and 5 are also close over the entire range of s and r. This may be explained by the fact that the number of groups $g = n = 10$ for model 2 and the systematic distribution for model 5 implicitly results in 10 groups. However, they do differ in the manner in which these groups are distributed.

  (iii) For $r \geq 0.95$, evaluations with all models, excepting model 4, are quite close.

  (iv) Evaluations with model 4 appear to significantly deviate from all other models for $r \geq 0.75$. This implies that modeling of the degree of replication is a very important task in availability evaluations.

- For transactions with majority-read/majority-write policy,

  (v) Evaluations with models 1 and 3 appear to be close. Similarly, evaluations with models 2 and 5 are close. In addition, evaluations with model 6 are close to evaluations with models 1 and 3.

  (vi) For $s \geq 25$, the availabilities appear to be independent of the read-set size. This implies that computations for $s > 25$ are redundant.

  (vii) The evaluations with models 2 and 5 seem to differ at higher values of n. The evaluations with the other four models are close for $n = 20$. This is an interesting observation.

  (viii) Once again, the variations in degree of replication of individual data objects appears to have a dominating effect on availability evaluations.

- For read-write transactions with read-one/write-all policy,

  (ix) The availabilities for $s \geq 5$ are significant only when $r \geq 0.99$.

(x) Since the availabilities are generally low, the effect of the differences in the models seem to be insignificant. At high reliabilities (i.e. $r \geq 0.99$), the evaluations with model 4 seem to deviate from the evaluations with the other models.

We will now study the effect of the individual model parameters.

- Models 1 and 3 are very simple, and need no further investigation.
- Evaluations with model 2 represent the effect of data object grouping on availability (Figure 4). As the number of groups is increased, the availability seems to be decreasing. This effect seems to diminish for $g \geq 25$. This effect is insignificant for read-write transactions. Similarly, this effect seems to vanish at high node reliabilities.
- Evaluations with model 4 represent the effect of variations in degrees of replication of data objects (Figure 5). The effect of these variations seem to be insignificant on read-write transactions. The effect of copy variations seem to be more apparent at high node reliabilities. Similarly, this effect seems to be more pronounced on read-only transactions (with read-one/write-all policy) than the other two classes.
- Model 5 represents the effect of data distribution on the availability evaluations. From Figure 6, it may be observed that the distribution effect is only evident at $s \geq 10$. In addition, the effects are more significant for read-only transactions than the other two classes. The effect is less evident at high node reliabilities.
- Model 6 represents the effect of node reliability variations on availabilities. From Figure 7, it may be observed that the variations have almost no effect on availability evaluations.

6 Conclusions

The current investigations on measuring the effect of data distribution, replication, and fault models on transaction availability evaluation have resulted in some very interesting observations. As part of this study, we chose six

models representing six different parametric assumptions that researchers and designers generally tend to make in their analysis. Using probabilistic analysis, we derived expressions for transaction availability for three classes of transactions: read-only (read-one/write-all policy), transactions with majority-read/majority-write policy, and read-write transactions (with read-one/write-all policy). The effect of the six parameters is measured by evaluating availabilities (for different read-set sizes). From here, we conclude that:

- By choosing a proper distributed database model, the computational complexity of transaction availability evaluations can be significantly reduced.
- For values of $s \leq 10$, all models result in almost the same transaction evaluation.
- It is not necessary to evaluate transaction availabilities for values of $s > 25$.
- Evaluations for the read-only transactions (with read-one/write-all policy) are more sensitive to database modeling than the other two classes of transactions.
- The degree of replication of individual (or group) data objects seems to have a significant effect on transaction availabilities. Thus, when different data objects have different copies, adopting average degree of replication to represent ant object in a system, may not result in accurate availability evaluations.
- The actual distribution of data object copies has some, if not significant, impact on availability evaluation.
- In a heterogeneous environment where different nodes may have different reliabilities, it is sufficient to represent each node by the average node reliability, without affecting the availability evaluations.
- Data object grouping (logical or physical) does not seem to effect the accuracy of availability evaluations as long as the number of groups is not too small (e.g. When $d = 1000$, $g \geq 25$ is sufficient).

Distributed database designers and researchers can utilize these results in choosing appropriate parameters that would result in reduced computational requirements without sacrificing the resulting accuracy of the design and analysis of these systems.

Appendix

Model 3 $< A_1, B_2, C_1, D_1, E_1 >$:
Here, we assume that each node has exactly the same number of data objects $(= \frac{d*c}{n})$.

$$TSA_s = \sum_{k=1}^{n} \binom{n}{k} r^k (1-r)^{n-k} \frac{\binom{x_k}{s}}{\binom{d}{s}} \tag{23}$$

$$TSA'_{x,y} = \sum_{k=1}^{n} \binom{n}{k} r^k (1-r)^{n-k} \frac{\binom{y_k}{y}}{\binom{d}{y}} \frac{\binom{x_k-y}{x}}{\binom{d-y}{x}} \tag{24}$$

$$TSA''_s = \sum_{k=m}^{n} \binom{n}{k} r^k (1-r)^{n-k} \frac{\binom{z_k}{s}}{\binom{d}{s}} \tag{25}$$

$$TFA_s = e^{-n_s t\lambda} \tag{26}$$

$$TFA'_{x,y} = e^{-n'_{x,y} t\lambda} \tag{27}$$

$$TFA''_s = e^{-n''_s t\lambda} \tag{28}$$

$$x_k = d\left[1 - \frac{\binom{n-k}{c}}{\binom{n}{c}}\right]$$

$$y_k = d\left[\frac{\binom{k}{c}}{\binom{n}{c}}\right]$$

$$z_k = d\sum_{l=m}^{c} \frac{\binom{k}{l}\binom{n-k}{c-l}}{\binom{n}{c}}$$

$$m = \left\lceil \frac{c+1}{2} \right\rceil$$

Model 4 $< A_1, B_1, C_2, D_1, E_1 >$:
Here, each data object may have its own degree of replication specified. For an efficient computation, we classify the data objects into p categories $(1 \leq p \leq n)$ based its degree of replication. d_l denoted the number of data objects in the l^{th} category where each object has c_l $(1 \leq c_l \leq n)$ copies.

$$TSA_s = \sum_{CA} Pr(CA) \sum_{k=1}^{\hat{n}} \binom{n}{k} r^k (1-r)^{n-k} \prod_{l=1}^{p} \left[1 - \frac{\binom{n-k}{c_l}}{\binom{n}{c_l}}\right]^{a_l} \tag{29}$$

$$TSA'_{x,y} = \sum_{CA'} Pr(CA') \sum_{CA''} Pr(CA'') \sum_{k=1}^{n} \binom{n}{k} r^k (1-r)^{n-k} \prod_{l=1}^{p} \left[1 - \frac{\binom{n-k}{c_l}}{\binom{n}{c_l}} \right]^{a'_l} \prod_{l=1}^{p} \left[\frac{\binom{k}{c_l}}{\binom{n}{c_l}} \right]^{a''_l} \quad (30)$$

$$TSA''_s = \sum_{CA} Pr(CA) \sum_{k=m}^{n} \binom{n}{k} r^k (1-r)^{n-k} \prod_{l=1}^{p} \left[\sum_{l'=m_l}^{c_l} \frac{\binom{k}{l'} \binom{n-k}{c_l - l'}}{\binom{n}{c_l}} \right]^{a_l} \quad (31)$$

$$CA = < a_1, a_2, \ldots, a_p >, \sum_{k=1}^{p} a_k = s, \forall k\ 1 \le k \le p\ 0 \le a_k \le d_k$$

$$CA' = < a'_1, a'_2, \ldots, a'_p >, \sum_{k=1}^{p} a'_k = x,\ \forall k\ 1 \le k \le p\ 0 \le a_k \le d_k$$

$$CA'' = < a''_1, a''_2, \ldots, a''_p >, \sum_{k=1}^{p} a''_k = y,\ \forall k\ 1 \le k \le p\ 0 \le a''_k \le d_k - a'_k$$

$$Pr(CA) = \frac{\binom{d_1}{a_1}\binom{d_2}{a_2}\cdots\binom{d_p}{a_p}}{\binom{d}{s}}$$

$$Pr(CA') = \frac{\binom{d_1}{a'_1}\binom{d_2}{a'_2}\cdots\binom{d_p}{a'_p}}{\binom{d}{x}}$$

$$Pr(CA'') = \frac{\binom{(d_1-a'_1)}{a''_1}\binom{d_2-a'_2}{a''_2}\cdots\binom{d_p-a'_p}{a''_p}}{\binom{d-x}{y}}$$

$$m_l = \left\lceil \frac{c_l + 1}{2} \right\rceil$$

The expressions for TFA_s, $TFA'_{x,y}$, and TFA''_s are the same as in Equations (26) - (28).

Model 5 $< A_1, B_1, C_1, D_2, E_1 >$:
Here, we assume that the entire data distribution is available as a dictionary, GD.

$$TSA_s = \sum_{S} Pr(S) \frac{\binom{f(S)}{s}}{\binom{d}{s}} \quad (32)$$

$$TSA'_{x,y} = \sum_{S} Pr(S) \frac{\binom{f'(S)}{y}}{\binom{d}{y}} \frac{\binom{f(S)-y}{x}}{\binom{d-y}{x}} \quad (33)$$

$$TSA''_s = \sum_S Pr(S) \frac{\binom{f''(S)}{s}}{\binom{d}{s}} \quad (34)$$
$$Pr(S) = r^{f'''(S)}(1-r)^{n-f'''(S)}$$

where
S - Node status vector; $S_j = 1 \Rightarrow$ Node j is up; $S_j = 0 \Rightarrow$ Node j is down.
$f(S)$ - The number of data objects available for read with the given node status vector (S). This is computed by scanning the columns of the GD matrix corresponding to the live nodes (as given by S).
$f'(S)$ - The number of data objects available for update (i.e. all c copies of these data objects are available at the live nodes) with the given node status vector (S). This is also computed by scanning the columns of the GD matrix corresponding to the live nodes (as given by S).
$f''(S)$ - The number of data objects available with a majority of copies among the available nodes. As before this is computed by scanning the columns of the GD matrix corresponding to the live nodes (as given by S).
$f'''(S)$ - The number of nodes available (or up) as indicated by the vector S.

Model 6 $< A_1, B_1, C_1, D_1, E_2 >$:
Here each node may have its own reliability. For computational purpose, we categorize the nodes based on their reliability. We assume that there are q $(1 \leq q \leq n)$ such categories. We let n_i to represent the number of nodes with reliability r_i, and a_i to represent the number of currently active (or up) nodes with this reliability.

$$TSA_s = \sum_{NA} Pr(NA) \left[1 - \frac{\binom{n-\sum_{i=1}^{q} a_i}{c}}{\binom{n}{c}}\right]^s \prod_{k=1}^{q} \left[\binom{n_k}{a_k} r_k^{a_k}(1-r_k)^{n_k-a_k}\right] \quad (35)$$

$$TSA'_{x,y} = \sum_{NA} Pr(NA) \left[\frac{\binom{\sum_{i=1}^{q} a_i}{c}}{\binom{n}{c}}\right]^y \left[1 - \frac{\binom{n-\sum_{i=1}^{q} a_i}{c}}{\binom{n}{c}}\right]^x \prod_{k=1}^{q} \left[\binom{n_k}{a_k} r_k^{a_k}(1-r_k)^{n_k-a_k}\right] \quad (36)$$

$$TSA''_s = \sum_{NA} Pr(NA) \left[\sum_{k=m}^{c} \frac{\binom{\sum_{i=1}^{q} a_i}{k}\binom{n-\sum_{i=1}^{q} a_i}{c-k}}{\binom{n}{c}}\right]^s$$

$$\prod_{k=1}^{q}\left[\binom{n_k}{a_k} r_k^{a_k}(1-r_k)^{n_k-a_k}\right] \tag{37}$$

$$Pr(NA) = \frac{\binom{n_1}{a_1}\binom{n_2}{a_2}\cdots\binom{n_q}{a_q}}{\binom{n}{\sum_{k=1}^{q} a_k}}$$

$$NA = < a_1, a_2, \ldots, a_q >,\ \forall i = 1, 2, \ldots, q\ 0 \leq a_i \leq n_i$$

$$\sum_{i=1}^{q} n_i = n$$

References

[1] M. Ahamad and M.H. Ammar, "Performance characterization of quorum-consensus algorithms for replicated data," *Tech. Report*, GIT-ICS-86/123, Georgia Institute of Technology, 1986.

[2] M.D. Beaudry, "Performance-related reliability measures for computing systems," *IEEE Trans. Computers*, Vol. C-27, pp. 540-547, June 1978.

[3] P.A. Bernstein and N. Goodman, "Concurrency control in distributed database systems," *ACM Computing Surveys*, Vol. 13, pp. 185-221, June 1981.

[4] B. Bhargava and L. Lilien, "A review of concurrency and reliability issues in distributed database systems," *Concurrency Control and Reliability Issues in Distributed Systems*, B. Bhargava (Ed.), Van Nostrand Reinhold Co, pp. 1-84, 1987.

[5] S. Ceri, G. Martella, and G. Pelagatti, "Optimal file allocation for a distributed database on a network of minicomputers," *Proc. International Conference on Data Bases*, University of Aberdeen, pp. 216-237, July 1980.

[6] E.G. Coffman, E. Gelenbe, and B. Plateau, "Optimization of number of copies in a distributed database," *IEEE Transactions on Software Engineering*, Vol. SE-7, No. 1, pp. 78-84, 1981.

[7] S.B. Davidson, "Analyzing partition failure protocols," *Technical Report*, MS-CIS-86-05, Dept. of Computer Science, University of Pennsylvania, January 1986.

[8] H. Garcia-Molina, "Performance evaluation of the update algorithms for replicated data in a distributed database," *Ph.D. Dissertation*, Computer Science Department, Stanford University, June 1979.

[9] H. Garcia-Molina and J. Kent, "Performance evaluation of reliable distributed systems," *Concurrency Control and Reliability in Distributed Systems*, B.K. Bhargava (Ed.), pp. 454-488, 1987.

[10] B. Gavish and H. Pirkul, "Computer and database location in distributed computer systems," *IEEE Transactions on Computers*, Vol. C-35, No. 7, pp. 583-590, July 1986.

[11] R. Mukkamala, "Design of partially replicated distributed database systems," *Technical Report,*, TR 87-04, Department of Computer Science, University of Iowa, July 1987.

[12] R. Mukkamala, S.C. Bruell, and R.K. Shultz, "Design of partially replicated distributed database systems: an integrated approach," *Proc. ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pp. 187-196, May 1988.

[13] R. Mukkamala, "Measuring the effect of data distribution and replication policies on performance evaluation of distributed database systems," *Proc. Fifth International Conference on Data Engineering*, February 1989.

[14] R. Mukkamala, "Measuring the effect of data replication and fault models on transaction availability analysis," *Technical Report*, TR 89-35, Department of Computer Science, Old Dominion University, May 1989.

[15] R. Mukkamala, "Performance evaluation of distributed database systems," *Technical Report*, TR 89-43, Department of Computer Science, Old Dominion University, June 1989.

[16] K.C. Sevcik, "Comparison of concurrency control methods using analytic methods," *Proc. Information Processing 83*, R.E.A. Mason (Ed.), North-Holland, September 1983.

[17] L.E. Stanfel, "Applications of clustering to information system design," *Information processing and Management*, Vol. 19, No. 1, pp. 37-50, 1983.

[18] R.B. Thomas, "A majority consensus approach to concurrency control for multiple copy databases," *ACM Transactions on Database Systems*, Vol. 4, No. 2, pp. 180-209, June 1979.

Figure 1a. n=10, d=1000, c=3, r=0.4

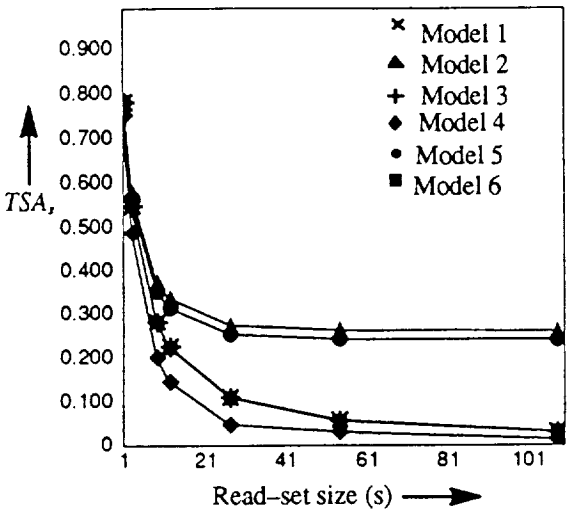

Figure 1b. n=10, d=1000, c=3, r=0.75

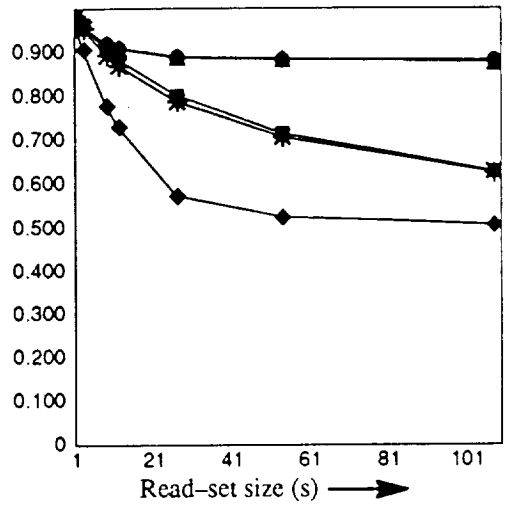

Figure 1c. n=10, d=1000, c=3, r=0.90

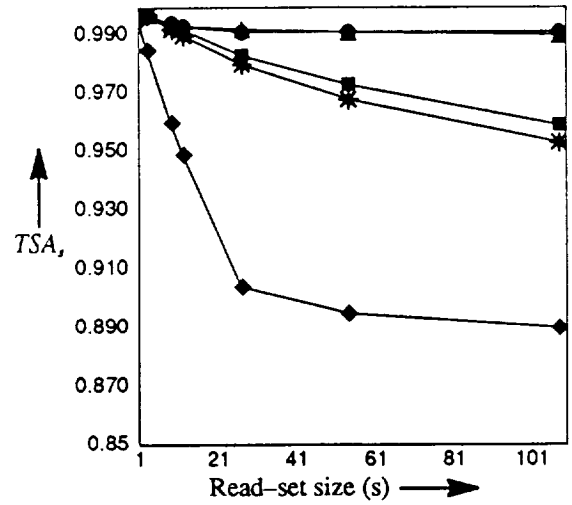

Figure 1d. n=10, d=10000, c=3, r=0.75

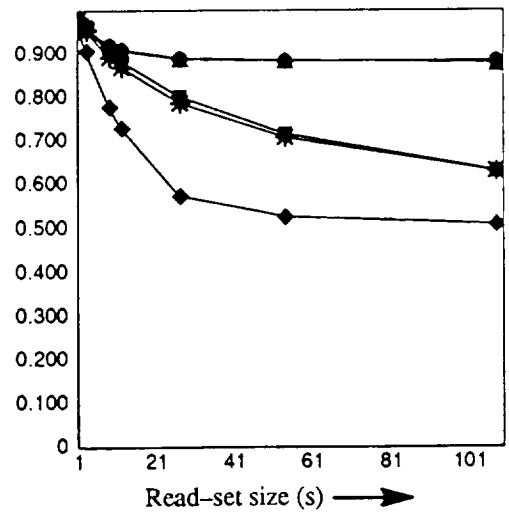

Figure 1. Transaction Start Availabilities for Read–Only Transactions (with Read–one/Write–all policy)

Figure 1e. n=10, d=1000, c=5, r=0.75

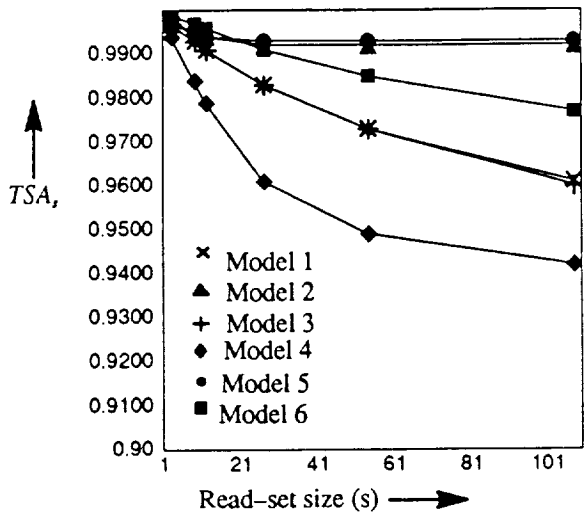

Figure 1f. n=20, d=1000, c=3, r=0.75

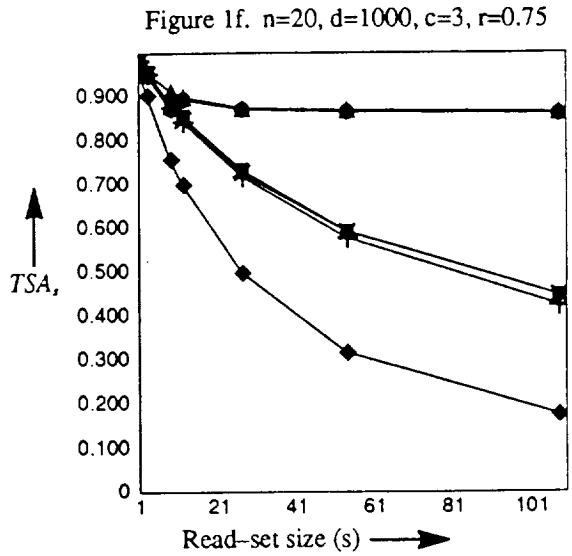

Figure 1g. n=10, d=1000, c=3, r=0.95

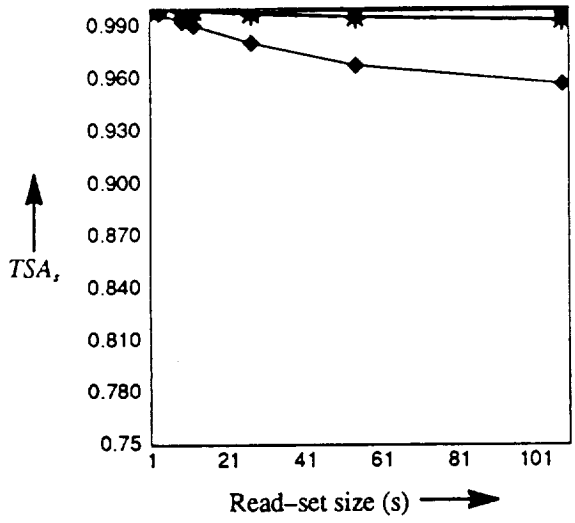

Figure 1 (Continued). Transaction Start Availabilities for Read–only Transactions (Read–one/Write–all Policy)

Figure 2a. n=10, d=1000, c=3, r=0.4

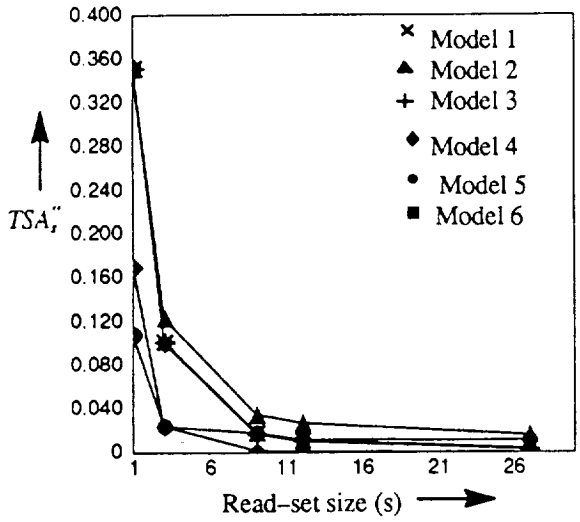

Figure 2b. n=10, d=1000, c=3, r=0.75

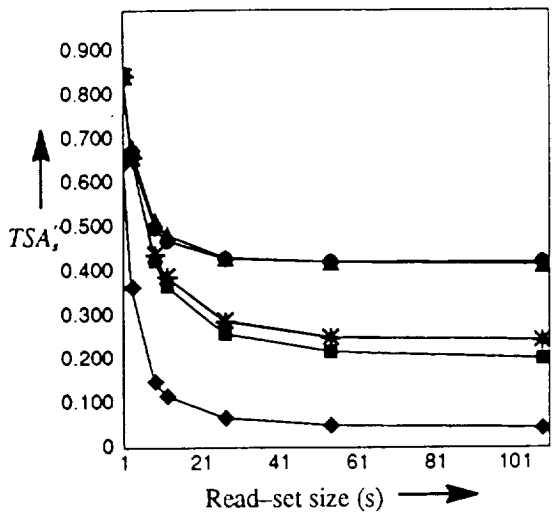

Figure 2c. n=10, d=1000, c=3, r=0.90

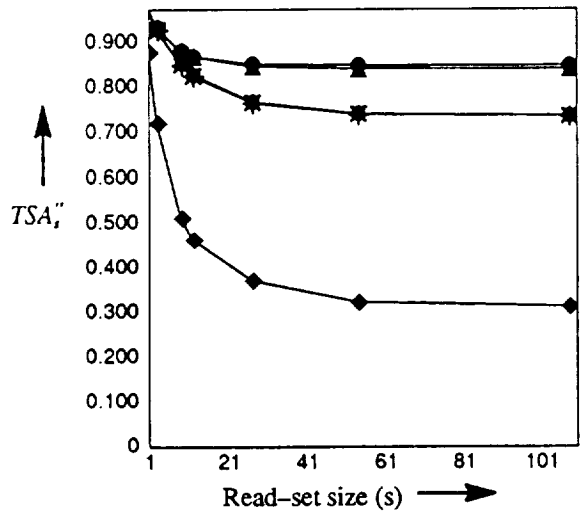

Figure 2d. n=10, d=10000, c=3, r=0.75

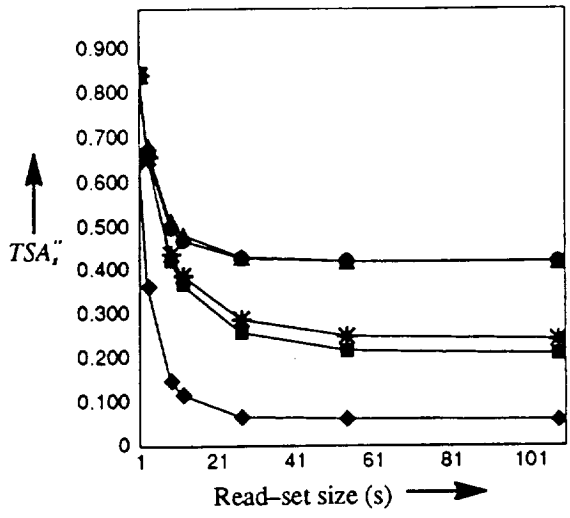

Figure 2. Transaction Start Availabilities with Read-Majority/Write-Majority Protocol

Figure 2e. n=10, d=1000, c=5, r=0.75

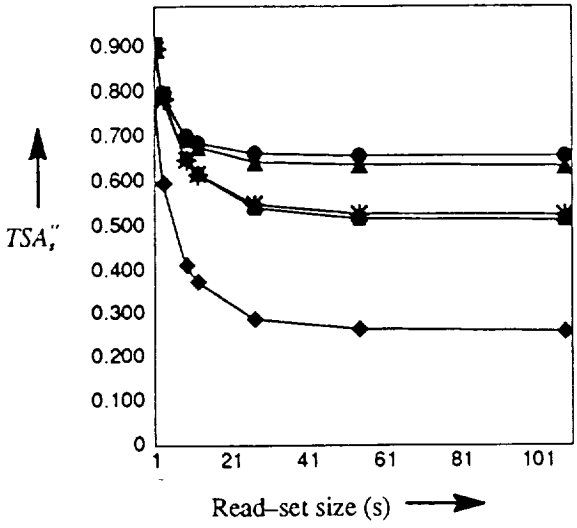

Figure 2f. n=20, d=1000, c=3, r=0.75

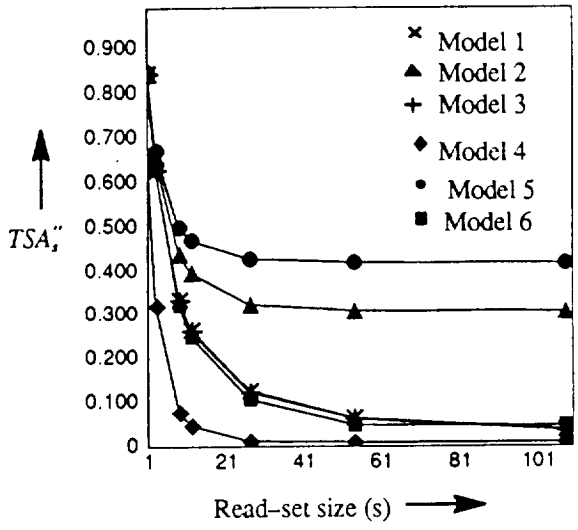

Figure 2g. n=10, d=1000, c=3, r=0.95

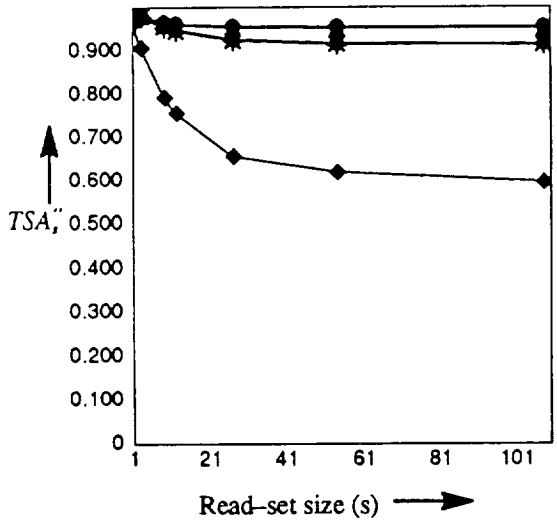

Figure 2 (Contd.). Transaction Start Availabilities with Read-Majority/Write-Majority Protocol

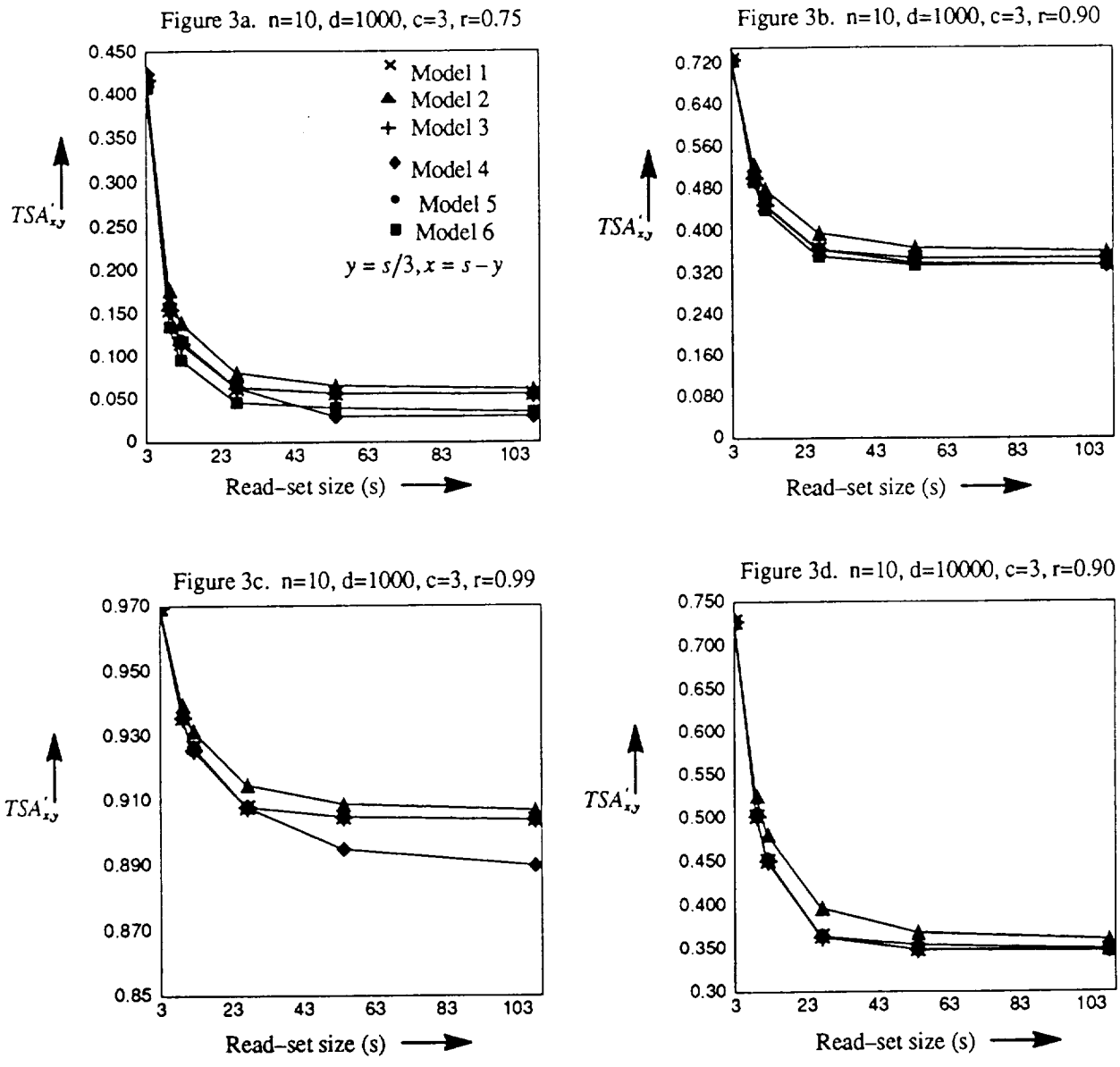

Figure 3. Transaction Start Availabilities for Read-write Transactions (with Read-one/Write-all Policy)

Figure 3e. n=10, d=1000, c=5, r=0.90

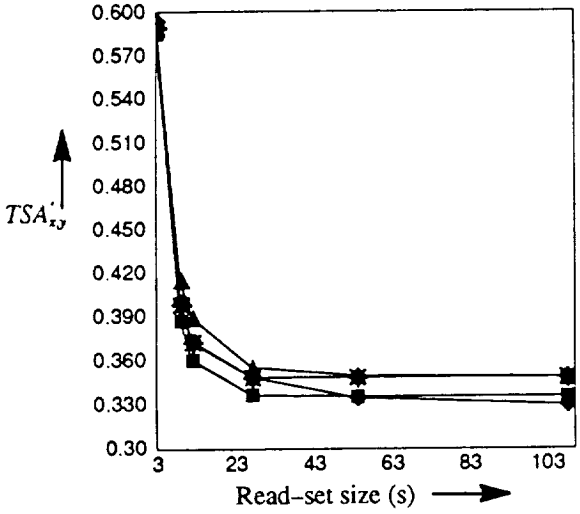

Figure 3f. n=20, d=1000, c=3, r=0.90

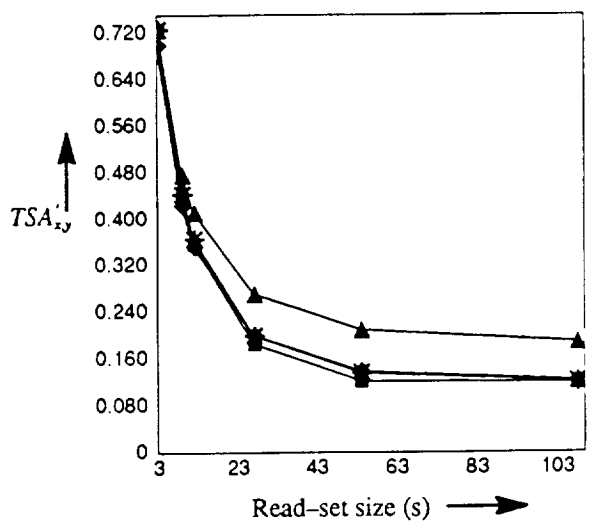

Figure 3 (Contd.) . Transaction Start Availabilities for Read-write Transactions (with Read-one/Write-all Policy)

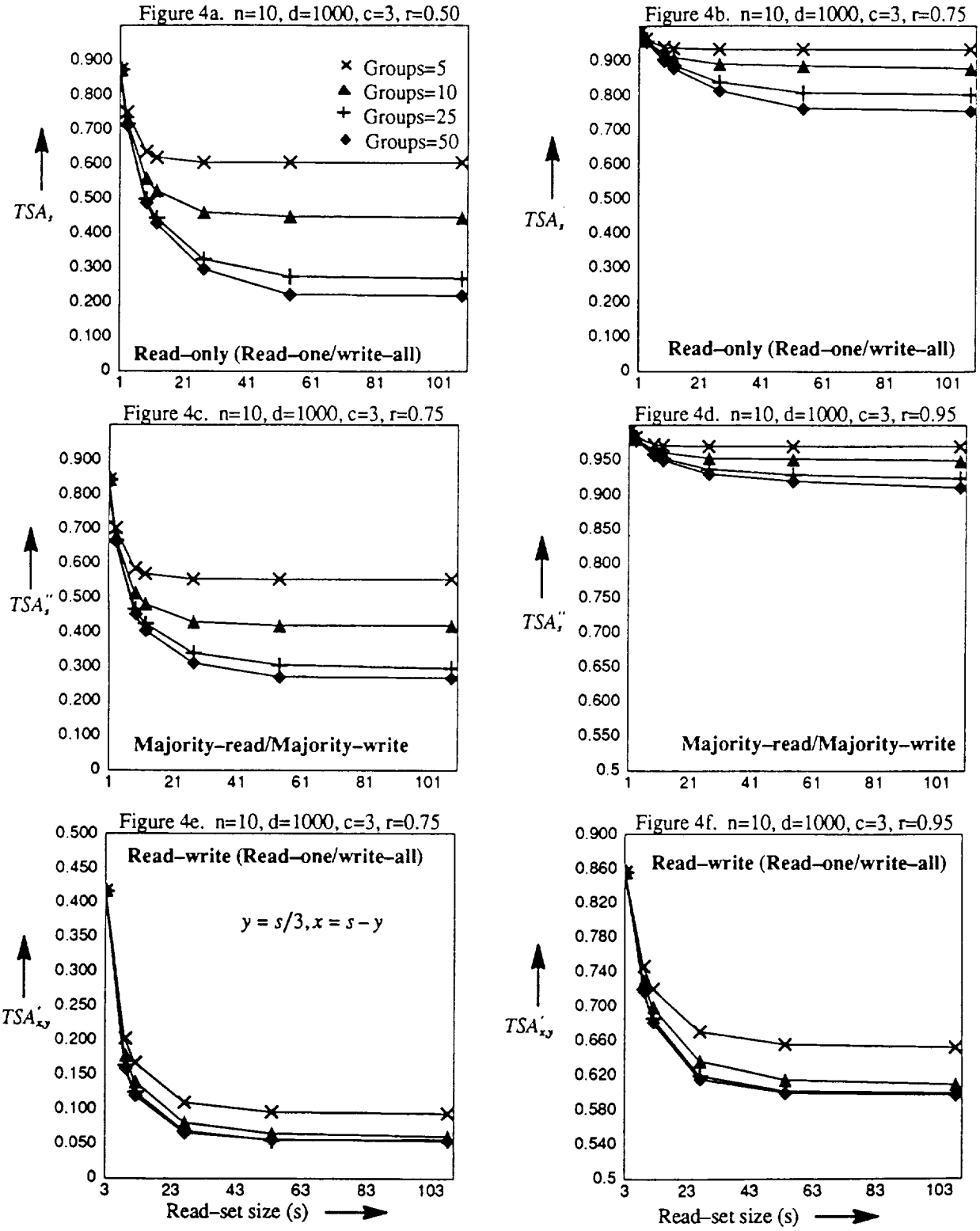

Figure 4. Illustration of the Effects of the Number of Groups on Availability Metrics (Model 2)

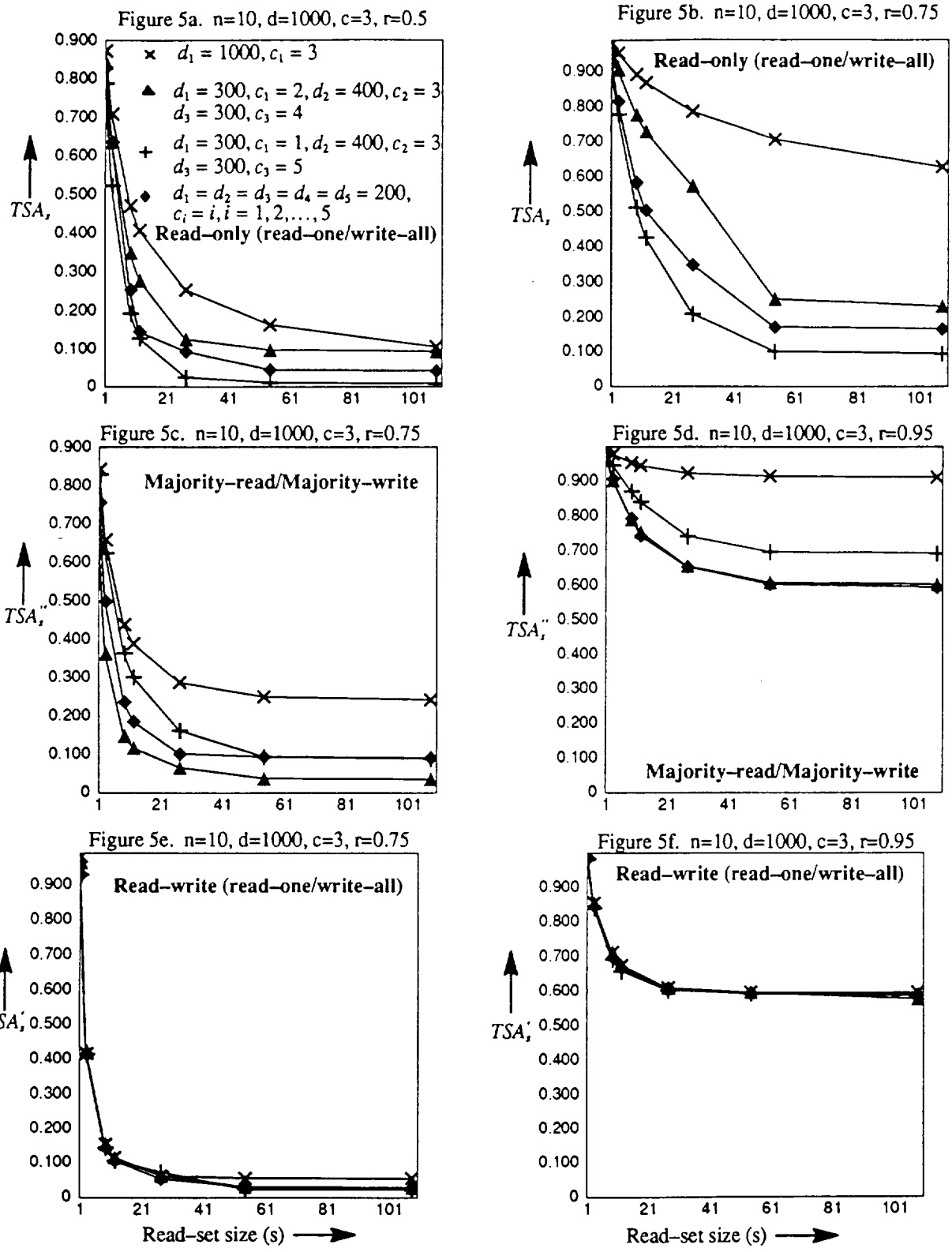

Figure 5. Illustration of the Effect of Copy variations on Availability (Model 4)

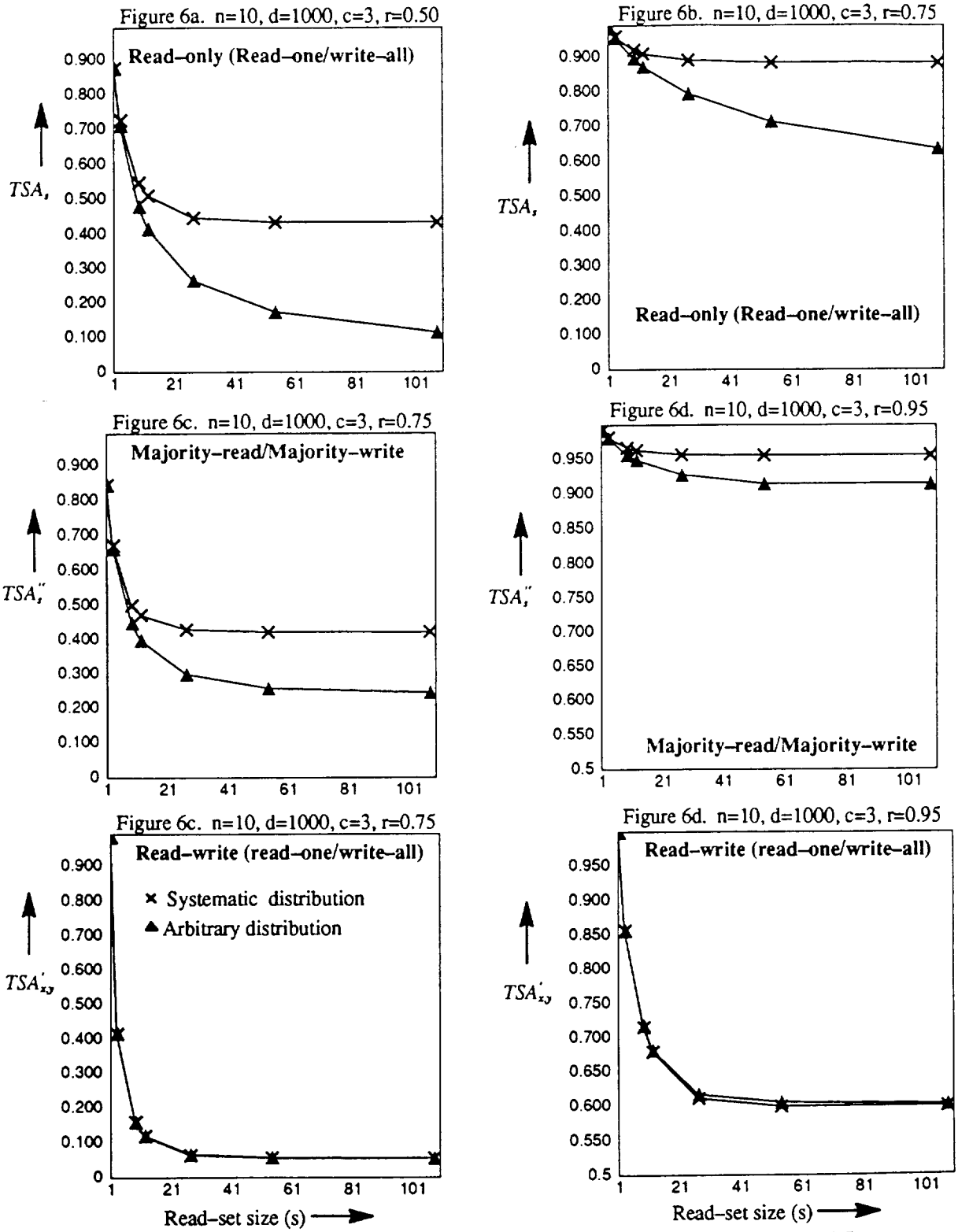

Figure 6. Illustration of the effect of Systematic Distribution on Availability (Model 5)

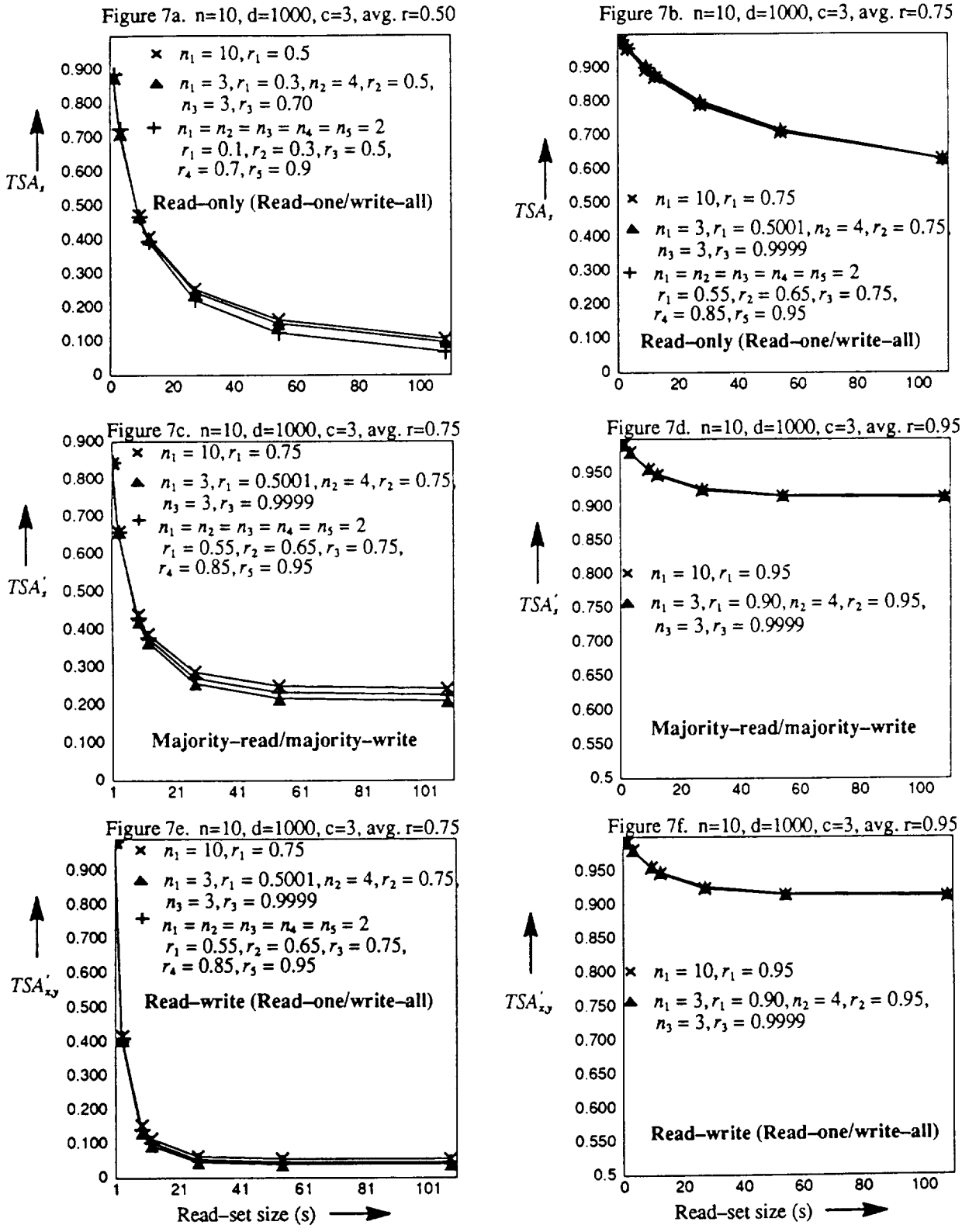

Figure 7. Illustration of the Effect of Reliability Variations on Availability (Model 6)

IEEE PROCEEDINGS OF THE

SOUTHEASTCON '91

NASA

Volume 2

91CH2998-3