Pergamon

Information Systems Vol. 20, No. 8, pp. 687–696, 1995
Copyright© 1995 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0306-4379/95 $9.50 + 0.00

0306-4379(95)00037-2

THE COLUMN PROTOCOL: A HIGH AVAILABILITY AND LOW MESSAGE COST SOLUTION FOR MANAGING REPLICATED DATA†

JEHN-RUEY JIANG

Management Information Systems Department, Chung Yuan Christian University, Chung Li, Taiwan, R. O. C.

(Received 18 January 1995; in final revised form 26 October 1995)

Abstract — This paper presents a quorum-based replica control protocol which is resilient to network partitioning. In the best case, the protocol generates quorums of a constant size. When some replicas are inaccessible, the quorum size increases gradually and may be as large as $O(n)$, where n is the number of replicas. However, the expected quorum size is shown to remain constant as n grows. This is a desirable property since the message cost for accessing replicated data is directly proportional to the quorum size. Moreover, the availability of the protocol is shown to be comparably high. With the two properties— constant expected quorum size and comparably high availability—the protocol is thus practical for managing replicated data.

Key words. Availability, Distributed Systems, Fault Tolerance, Replication, Quorums.

1. INTRODUCTION

In a distributed system, data can be replicated at different sites to tolerate site and/or network link failures. However, complex replica control protocols are required to make multiple replicas of a data object behave as a single one, i.e., to ensure *one-copy equivalence* [4]. Several replica control protocols [1, 2, 3, 5, 8, 9, 10, 11, 12] have been developed on the basis of quorum consensus concept, which is described below. Each replica is associated with a *version number*. A read operation should *read-lock* and access a read quorum of replicas and return the replica owning the largest version number. On the other hand, a write operation should *write-lock* and access a write quorum of replicas and then updates them with the new version number which is one more than the largest version number just encountered. To ensure that a read operation can always return the most up-to-date replica, any pair of a read and a write quorum and any two write quorums are required to have a non-empty intersection. The quorum-based protocols are fault- tolerant in the sense that even when network partitioning [6] occurs and makes some replicas inaccessible, quorums may still be formed successfully.

In this paper, we propose a quorum-based replica control protocol, named *column protocol*. With the aid of a logical structure called *multi-column*, the column protocol achieves $O(1)$ (constant) read and write quorum sizes in the best case. When some replicas are inaccessible, the quorum size increases gradually and may be as large as $O(n)$, where n is the number of replicas. However, the expected quorum size is shown to remain constant as n grows. This is a desirable property since the message cost for accessing replicated data is directly proportional to the quorum size. In addition, the availability of the column protocol is shown to be comparably high. With the two properties—constant expected quorum size and comparably high availability—the column protocol is thus practical for managing replicated data.

The remainder of this paper is organized as follows. In Section 2, we describe the column protocol in detail. In Section 3, we show the correctness of the column protocol. In Section 4, we analyze the column protocol and compare it with related ones in terms of quorum size and availability. Finally, we conclude this paper with Section 5.

†Recommended by Amr El Abbadi

2. THE COLUMN PROTOCOL

In this section, we describe the column protocol. We first introduce the multi-column structure. Then, we define the read and write quorums under the assumption that data replicas are organized as a multi-column structure. At last, two functions that can properly generate the defined quorums are proposed as the skeleton for the protocol implementation.

A *multi-column structure* $MC(k) = (C_1, \ldots, C_k)$ is a list of pairwise disjoint sets of replicas. Each set C_i is called a *column* and must satisfy $|C_i| > 1$ for $1 \leq i \leq k$ (the necessity for this restriction will be explained later). For example, $(\{u_1, u_2\}, \{u_3, u_4, u_5\}, \{u_6, u_7, u_8, u_9\})$ and $(\{u_1, u_2, u_3, u_4, u_5\}, \{u_6, u_7\}, \{u_8, u_9\})$, with $u_1, \ldots, u_9$ being replicas, are multi-column structures.

By organizing data replicas as multi-column structure $MC(k) = (C_1, \ldots, C_k)$, we define the write and read quorums as follows:

A *write quorum* under $MC(k)$ is a set that contains all replicas of some column $C_i, 1 \leq i \leq k$ (note that $i = 1$ is included), and one replica of each of the columns $C_{i+1}, \ldots, C_k$.

A *read quorum* under $MC(k)$ is either
Type-1: a set that contains one replica of each of the columns $C_1, \ldots, C_k$.
or
Type-2: a set that contains all replicas of some column $C_i, 1 < i \leq k$ (note that $i = 1$ is excluded), and one replica of each of the columns $C_{i+1}, \ldots, C_k$.

We now explain the necessity of the restriction $|C_i| > 1, 1 \leq i \leq k$. It is used to make the following two cases distinguishable: (case 1) a set contains one replica of C_i and (case 2) a set contains all replicas of C_i.

Below, we show an example of quorum construction. Under $MC(2) = (\{u_1, u_2, u_3\}, \{u_4, u_5\})$, the possible write quorums are $\{u_4, u_5\}$, $\{u_1, u_2, u_3, u_4\}$, $\{u_1, u_2, u_3, u_5\}$, and the possible read quorums are $\{u_1, u_4\}$, $\{u_1, u_5\}$, $\{u_2, u_4\}$, $\{u_2, u_5\}$, $\{u_3, u_4\}$, $\{u_3, u_5\}$ (of type-1) and $\{u_4, u_5\}$ (of type-2). Note that the write quorum definition and the type-2 read quorum definition are identical except that the latter does not include the sets composed of all replicas in C_1 and one replica from each of $C_2, \ldots, C_k$. For the sack of efficiency, the sets mentioned are not regarded as read quorums because each of them is a super set of a type-1 read quorum.

In an extreme case, all replicas in C_k can constitute a quorum that is of a constant size $|C_k|$. And in another extreme case, one replica from each of $C_1, \ldots, C_k$ (for a type-1 read quorum) or all replicas in C_1 together with one replica from each of $C_2, \ldots, C_k$ (for a write quorum) can constitute a quorum. If the size of each $C_i, 1 \leq i \leq k$, is constant (or bounded above and below by a constant), then the quorum mentioned is of size $O(n)$.

Below we introduce two functions, *Get_Write_Quorum* and *Get_Read_Quorum*, which can properly return a write quorum and a read quorum under $MC(k)$, respectively. Note that we assume $wlock(C_i)$ is a function that tries to write-lock and return replicas of C_i. It locks and returns (case 1) the set of all replicas of C_i if they are all lockable, or (case 2) a singleton set of one arbitrary lockable replica if more than one replica is lockable, or (case 3) an empty set, otherwise. Note that when $wlock(C_1)(i = 1)$ is performed, (case 2) is ruled out, i.e., either the set of all replicas of C_1 or an empty set is returned. Function $rlock(C_i)$ is identical to $wlock(C_i)$ except that $rlock(C_i)$ uses read-lock instead of write-lock and that when $rlock(C_1)$ is performed, (case 1) is ruled out, i.e., either a singleton set of one lockable replica of C_1 or an empty set is returned.

Function *Get_Write_Quorum* $(MC(k) = (C_1, \ldots, C_k)$: **Multi-Column**): **Set**;
Var S: **Set**;
    $S = wlock(C_k)$;
    If $S = C_k$ **Then** *Return*(S);
    If $|S| = 1$ **Then** *Return*$(S \cup$ *Get_Write_Quorum*$(MC(k-1) = (C_1, \ldots, C_{k-1}))$;
    If $S = \emptyset$ **Then** *Exit*(**failure**); //Unsuccessful in forming a write quorum//
End *Get_Write_Quorum*

Function *Get_Read_Quorum* ($MC(k) = (C_1, \ldots, C_k)$: **Multi-Column**):**Set**;
Var S: **Set**;
$S = rlock(C_k)$;
If $S = C_k$ **Then** *Return*(S);
If $|S| = 1$ **and** $k > 1$ **Then** *Return*($S \cup Get\_Read\_Quorum(MC(k-1) = (C_1, \ldots, C_{k-1}))$);
If $|S| = 1$ **and** $k = 1$ **Then** *Return*(S);
If $S = \emptyset$ **Then** *Exit*(**failure**); //Unsuccessful in forming a read quorum//
End *Get_Read_Quorum*

3. CORRECTNESS

In this section, we prove the correctness of the column protocol with the following lemmas. Lemma 1 is concerned with the write-write intersection property (i.e., any pair of write quorums having a non-empty intersection), while Lemmas 2 and 3 are related to read-write intersection property (i.e., any pair of a read quorum and a write quorum having a non-empty intersection).

Lemma 1 *Any two write quorums intersect.*

Proof. Under $MC(k) = (C_1, \ldots, C_k)$, let Q_1 be a write quorum containing all replicas of C_i and one replica from each of $C_{i+1}, \ldots, C_k$, and let Q_2 be a write quorum containing all replicas of C_j and one replica from each of $C_{j+1}, \ldots, C_k$. Without loss of generality, we may assume $i > j$ (the proof for the case of $i = j$ is trivial and omitted). Q_1 and Q_2 must intersect since Q_1 contains all replicas of C_i and Q_2 contains one replica of C_i. □

Lemma 2 *A type-1 read quorum and a write quorum intersect.*

Proof. Because a type-1 read quorum contains one replica from each column and a write quorum contains all replicas of a certain column, the two quorums must intersect. □

Lemma 3 *A type-2 read quorum and a write quorum intersect.*

Proof. Since a type-2 read quorum is of the same form of a write quorum, the proof is similar to that of Lemma 1 and omitted. □

4. ANALYSIS AND COMPARISON

In this section we first analyze the quorum size and the availability for the column protocol, and then compare the analyzed results with those of related protocols. In the following analysis, we assume that all data replicas have the same *up-probability*, p, the probability that a single replica is up (i.e., accessible). We also assume that replicas are organized as $MC(k) = (C_1, \ldots, C_k)$ and use S_i to denote $|C_i|$, for $1 \leq i \leq k$.

4.1. Availability

The read (respectively, write) availability is defined to be the probability of a read (respectively, write) quorum being successfully formed in an error-prone environment. For $k > 1$, if all replicas in C_k are up, then a read (or write) quorum under $MC(k)$ can be formed. On the other hand, if at least one replica but not all replicas in C_k are up, then one of the up replicas together with a read (respectively, write) quorum under $MC(k-1)$ can form a read (respectively, write) quorum under $MC(k)$. Let $R_{AV}(k)$ denote the availability of read quorums under $MC(k)$, and $W_{AV}(k)$, the availability of write quorums under $MC(k)$. For $k > 1$, we have

$$\begin{aligned} R_{AV}(k) &= Prob.(\text{all replicas in } C_k \text{ are up}) + \\ &\quad Prob.(\text{at least one replica but not all replicas in } C_k \text{ are up}) \times R_{AV}(k-1) \\ &= p^{S_k} + (1 - p^{S_k} - (1-p)^{S_k}) R_{AV}(k-1) \end{aligned} \tag{1}$$

$$\begin{aligned} W_{AV}(k) &= Prob.(\text{all replicas in } C_k \text{ are up}) + \\ &\quad Prob.(\text{at least one replica but not all replicas in } C_k \text{ are up}) \times W_{AV}(k-1) \\ &= p^{S_k} + (1 - p^{S_k} - (1-p)^{S_k}) W_{AV}(k-1) \end{aligned} \tag{2}$$

For $k = 1$, if at least one replica in C_1 is up, then a read quorum under $MC(1)$ can be formed. And all replicas in C_1 are required to be up to form a write quorum under $MC(1)$. Thus, we have $R_{AV}(1) = (1 - (1-p)^{S_1})$ and $W_{AV}(1) = p^{S_1}$.

A fixed number of replicas can be arranged as a variety of multi-column structures. To reduce the number of analysis cases, we limit all columns to have the same size s; that is, we assume $|C_1| = \cdots = |C_k| = s$ (i.e., $S_1 = \cdots = S_k = s$) for $MC(k) = (C_1, \ldots, C_k)$. Below, we use $MC(k, s)$ to denote such a multi-column structure.

When $MC(k, s)$ is considered, the recursive equations (1) and (2) can be regarded as first-order linear difference equations [7]†, which can be solved analytically. We have

$$R_{AV}(k) = (1 - p^s - (1-p)^s)^{k-1} \left(1 - (1-p)^s - \frac{p^s}{p^s + (1-p)^s}\right) + \frac{p^s}{p^s + (1-p)^s} \tag{3}$$

$$W_{AV}(k) = (1 - p^s - (1-p)^s)^{k-1} \left(p^s - \frac{p^s}{p^s + (1-p)^s}\right) + \frac{p^s}{p^s + (1-p)^s} \tag{4}$$

We first apply equations (3) and (4) to investigate the influence of column sizes under a fixed number of replicas. We consider the following multi-column structures for a 30-replica system: $MC(15, 2), MC(10, 3), MC(6, 5), MC(5, 6), MC(3, 10)$ and $MC(2, 15)$. The read availabilities corresponding to those structures are depicted in Figure 1, which reveals that larger column sizes usually render the read availability higher (because they make the construction of type-1 read quorums easier). The write availabilities corresponding to those structures are depicted in Figure 2, which reveals that smaller column sizes usually render the write availability higher (because they make the construction of write quorums easier).

There are trade-offs between the read availability and the write availability. However, one can choose a proper column size according to practical situations, such as the fractions of read and write operations, and the constraints on the lowest read or write availabilities, etc. Since the read availabilities are on the upper side and the write availabilities are on the lower side, we suggest adopting small column sizes, say 3 or 5, so that both the read and write availabilities are comparably high. We do not suggest adopting the column size of 2, which leads to lower write availabilities than those resulting from sizes of 3 and 5 for large up-probability p (e.g., for $p > 0.75$). Note that most practical systems have large up-probability p, under which the column size of 2 causes a relatively large probability of no replica in a column being up, which prohibits the construction of any quorum.

We now apply equations (3) and (4) to investigate the asymptotic value of quorum availability. When k goes to infinity, the term $(1 - p^s - (1-p)^s)^{k-1}$ goes to 0, and both $R_{AV}(k)$ and $W_{AV}(k)$ go to

$$\frac{p^s}{p^s + (1-p)^s} = \frac{1}{1 + \left(\frac{1-p}{p}\right)^s}.$$

In other words, the asymptotic availability of the column protocol is $\dfrac{1}{1 + \left(\frac{1-p}{p}\right)^s}$. For $p = 0.5$, the asymptotic availability is 0.5 whatever the column size is. For $p < 0.5$, $\frac{1-p}{p}$ is larger than 1 and

† A first-order linear difference equation of the form $X_k = aX_{k-1} + b$ for $k \geq 2$ with X_1 being the first term has as its kth term $X_k = a^{k-1}(X_1 + \frac{b}{a-1}) - \frac{b}{a-1}$ if $a \neq 1$.

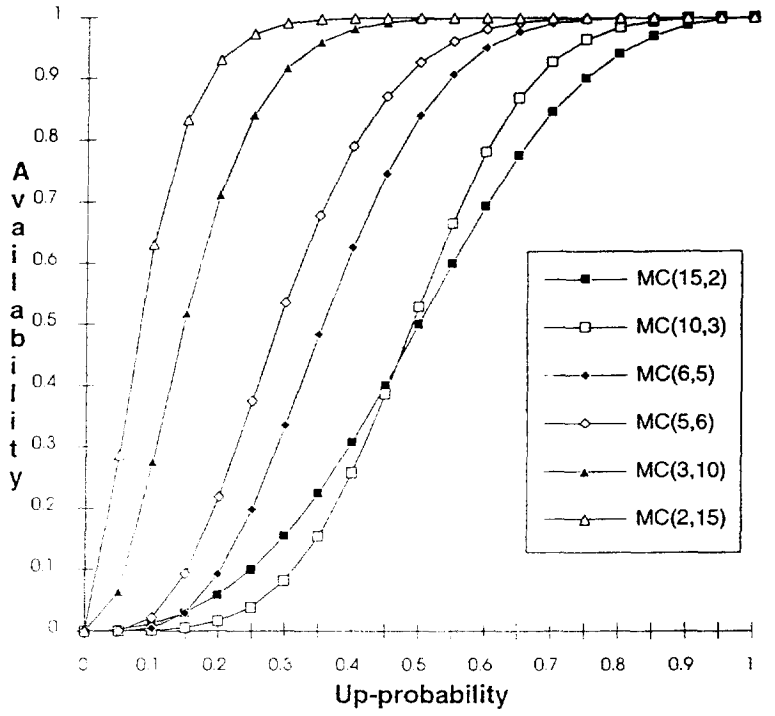

Fig. 1: The read availability of the column protocol for various multi-column structures.

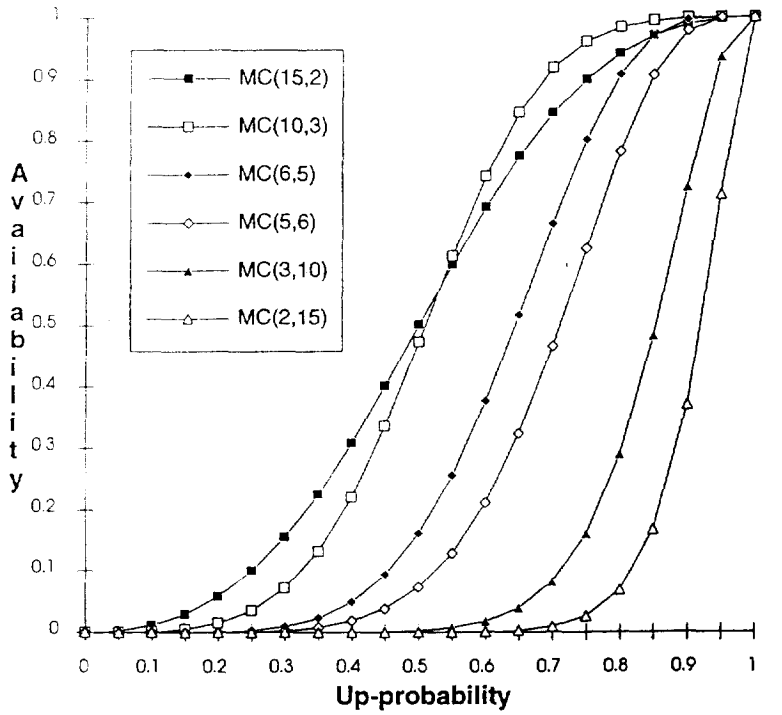

Fig. 2: The write availability of the column protocol for various multi-column structures.

thus $(\frac{1-p}{p})^s$ increases as s grows. It is easy to see that the smaller s is, the larger the asymptotic availability is. For $p > 0.5$, $\frac{1-p}{p}$ is less than 1 and thus $(\frac{1-p}{p})^s$ decreases as s grows. It is easy to see that the larger s is, the larger the asymptotic availability is. To sum up, smaller column sizes are preferable when $p < 0.5$ and larger column sizes are preferable when $p > 0.5$. However, we still suggest adopting small column sizes because the asymptotic availability is high even for small column sizes when $p > 0.5$. For example, when $s = 3$, the asymptotic availability is 0.998630, 0.984615 and 0.927027 for p=0.9, 0.8 and 0.7, respectively. When $s = 4$, the asymptotic availability is 0.999847, 0.996108 and 0.967365 for p=0.9, 0.8 and 0.7, respectively.

4.2. Quorum Size

Both the smallest read and write quorums under $MC(k, s), k >> s$, are of size s; such quorums are formed by including only all replicas in the last column. That is, the column protocol has constant quorum size in the best case. This is a desirable property since the message cost for accessing replicated data is directly proportional to the quorum size. However, when some replicas are inaccessible, the quorum size increases gradually and may be as large as $O(n)$. Specifically, under $MC(k, s), k >> s$, the largest read quorum, which is composed of one replica from each of $C_1, \ldots, C_k$, is of size $k = \frac{n}{s}$. And the largest write quorum, which is composed of all replicas from C_1 and one replica from each of $C_2, \ldots, C_k$, is of size $s + k - 1 = s + \frac{n}{s} - 1$. Note that both quorums are of $O(n)$ sizes.

Using the lower and the upper quorum size bounds to estimate data access cost may be too optimistic and too pessimistic respectively. Below, we analyze the expected quorum size for the column protocol to estimate its average data access cost. Let $R_{ES}(k)$ and $W_{ES}(k)$ denote respectively the expected sizes of read and write quorums under $MC(k)$. We apply parameter f, which is also adopted in the tree quorum protocol [2], to indicate the fraction of the quorums composed of only all the replicas in C_k (note that f is used in the tree quorum protocol [2] to indicate the fraction of quorums including the root node). For $k > 1$, we have

$$R_{ES}(k) = fS_k + (1-f)(1 + R_{ES}(k-1)) = (fS_k + 1 - f) + (1-f)R_{ES}(k-1) \quad (5)$$

$$W_{ES}(k) = fS_k + (1-f)(1 + W_{ES}(k-1)) = (fS_k + 1 - f) + (1-f)W_{ES}(k-1) \quad (6)$$

The term fS_k arises because there are f quorums of size S_k that are composed of only all replicas in C_k. And the term $(1-f)(1 + R_{ES}(k-1))$ (respectively, $(1-f)(1 + W_{ES}(k-1))$) arises because there are $(1-f)$ quorums of size $R_{ES}(k-1)+1$ (respectively, $W_{ES}(k-1)+1$) that are composed of not all replicas of C_k, but one replica of C_k and one quorum under $MC(k-1)$. Since one arbitrary replica of C_1 can form a read quorum under $MC(1)$, and all replicas in C_1 can form a write quorum under $MC(1)$, we have $R_{ES}(1) = 1$ and $W_{ES}(1) = S_1$.

When $MC(k, s), k >> s$, is considered, the case of $f = 1$ corresponds to the lower bound of the quorum size, which occurs when all the replicas in C_k are always included in the quorum. On the other hand, the case of $f = 0$ corresponds to the upper bound of the quorum size, which occurs when a larger quorum is always chosen instead of a smaller one. Note that the probability that all replicas in C_k are up (i.e., p^s) can reflect the value of f. For example, the value of f can be reflected by 0.65^3=0.274625 when p=0.65 and s=3.

Under $MC(k, s)$ where $S_1 = \cdots = S_k = s$, the recursive equations (5) and (6) can be regarded as first-order linear difference equations and can be solved analytically. For $f > 0$, we have

$$R_{ES}(k) = (1-f)^{k-1}(1 - \frac{fs + 1 - f}{f}) + \frac{fs + 1 - f}{f} \quad (7)$$

$$W_{ES}(k) = (1-f)^{k-1}(s - \frac{fs + 1 - f}{f}) + \frac{fs + 1 - f}{f} \quad (8)$$

When k goes to infinity (and so does n), the term $(1-f)^{k-1}$ goes to 0, and hence both $R_{ES}(k)$ and $W_{ES}(k)$ go to $\frac{fs+1-f}{f} = s + \frac{1}{f} - 1$, which is a constant. In other words, the expected quorum size of the column protocol remains constant when n grows. It is easy to see that smaller s or larger f produces smaller asymptotic expected quorum size. Take the following four cases for example:

(case 1) $f = 0.5, s = 3$ (case 2) $f = 0.5, s = 5$ (case 3) $f = 0.25, s = 3$ and (case 4) $f = 0.25, s = 5$. The asymptotic expected quorum sizes for these four cases are 4, 6, 6 and 8 , respectively.

4.3. Comparison

In this section we describe some related protocols [2, 3, 5, 8, 9, 10, 11, 12] and compare them with the column protocol (CP) in terms of availability and quorum size. The simplest replica control scheme is the read- one-write-all protocol (ROWA) [3], in which any replica can form a read quorum and all the replicas can form a write quorum. ROWA can be regarded as a special case of CP—when we apply the multi-column structure with only one column containing all the replicas. The majority quorum protocol (MQP) [12] requires both the read and the write quorums to have over half (i.e., at least $\lceil \frac{n+1}{2} \rceil$) replicas; thus, its quorum size is $O(n)$.

Some protocols [2, 8] form quorums with the aid of tree structures. By placing replicas in leaves of a mutilevel tree with non-leaf nodes being logical, the hierarchical quorum protocol (HQP) [8] achieves $O(n^{0.63})$ quorum size. Its quorum forming is hierarchical: a quorum of a node at level i is formed if enough (over half) quorums of its child nodes at level $i + 1$ are formed. Thus, any two quorums formed at the root have a non-empty intersection and can be used as a write (or read) quorum. Assuming replicas are logically organized as a binary tree, the tree quorum protocol (TQP) [2] has $\lceil \log n \rceil$ quorum size in the best case. Its quorum forming (for both read and write quorums) is recursive and can be regarded as attempting to obtain replicas from nodes along a root-to-leaf path. If the root fails, then the obtaining should follow two paths: one root-to-leaf path on the left subtree and one root-to-leaf path on the right subtree. The largest quorum of TQP is composed of all leaf nodes and is of size $\lceil \frac{n+1}{2} \rceil$; however, it has been shown in [2] that TQP has $O(\log n)$ quorum size for most practical environments.

In the grid protocol (GP) [5], replicas are organized as a rectangular grid of l rows and m columns, where $l \times m = n$ (the number of replicas). A *column-cover*, which contains one replica of each column, can form a read quorum, and a column-cover along with all replicas of some column can form a write quorum. Thus, a read quorum contains m replicas and a write quorum contains $l + m - 1$ replicas. If a square grid is assumed, i.e., $l = m = \sqrt{n}$, then both the read and write quorums have $O(\sqrt{n})$ size.

GP is dominated by CP, which means that if a quorum can be formed in GP then a quorum can be formed in CP. Below, we verify the last statement. Consider the multi-column structure $MC(k, s)$, which is exactly a k-column, s-row grid structure. Under such a structure, a write quorum of GP is a super set of some write quorum under $MC(k, s)$ (by the definitions of the two quorums discussed), and a read quorum of GP is actually a type-1 read quorum under $MC(k, s)$ (and CP still has type-2 read quorums). Therefore, we can conclude that GP is dominated by CP.

In the hierarchical grid protocol (HGP) [9], a hierarchical grid structure is used in which nodes at the lowest level 0 are physical replicas and nodes at level $i (i > 0)$ are defined as a square grid of level $i - 1$ nodes. The quorum forming is recursive and is identical to that of GP if viewed at a single level. The read (respectively, write) quorum formed at the top level allows a read (respectively, write) operation to proceed. If a square grid structure is assumed in each level, the hierarchical grid protocol also owns $O(\sqrt{n})$ quorum size for both write and read quorums. HGP has the property that its quorum availability increases asymptotically when more replicas are used, a property not owned by GP. Since HGP bases on quorum definitions of GP, it can be improved by replacing GP's quorum definitions with CP's.

The general grid protocol (GGP) [10] improves GP by regarding either a column-cover or a full column of replicas as a read quorum (this improvement was also suggested independently by Neilsen [11]) and by allowing empty (hollow) grid positions that correspond to no data replica. It has been shown in [10] that empty grid positions usually make quorum availability higher. Note that GGP has the same write quorum size as GP and any GGP's write quorum is a super set of some CP's write quorum. However, any CP's read quorum is a super set of some GGP's read quorum.

A summary of quorum sizes of some of the discussed protocols appears in Table 1. Availability comparisons of CP, ROWA, MQP and TQP for 15- and 31-replica systems appear in Figures 3

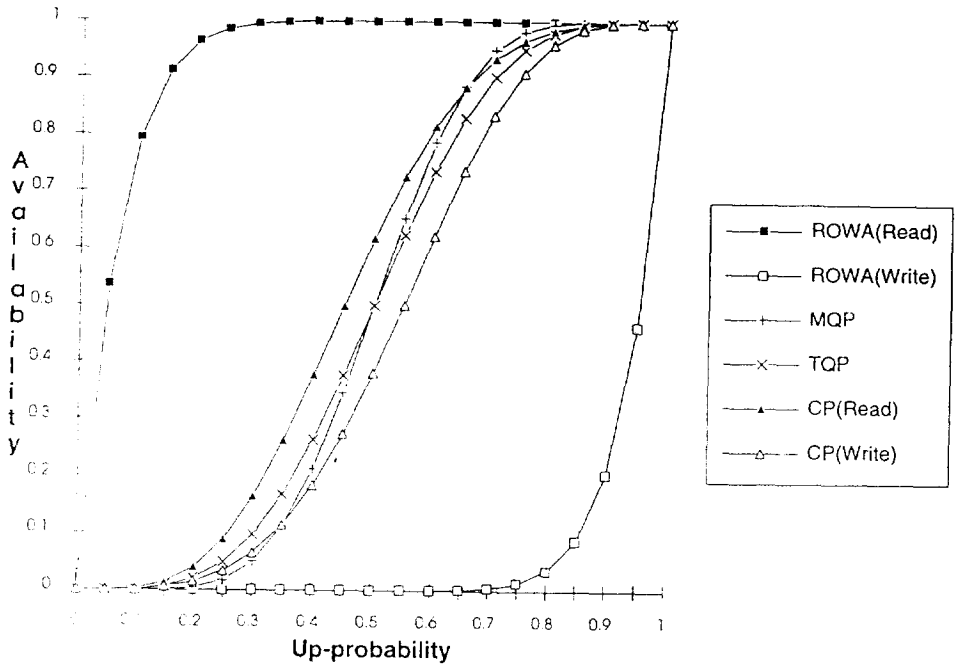

Fig. 3. The availability comparison for various protocols for the 15-replica system.

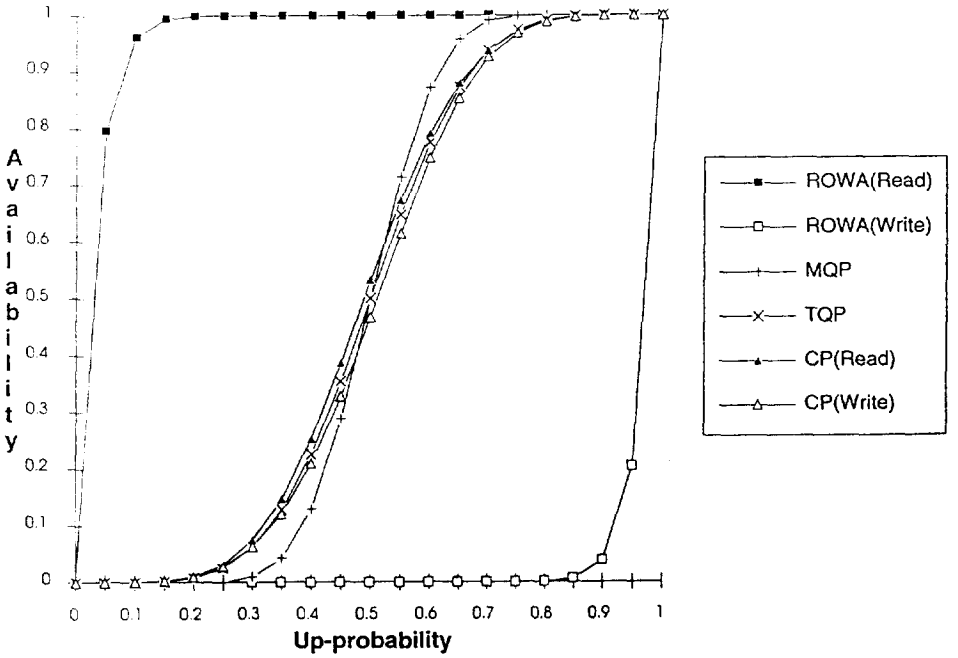

Fig. 4: The availability comparison for various protocols for the 31-replica system.

	MQP	HQP	TQP	GP	HGP	CP
Lower Bound	$\lceil \frac{n+1}{2} \rceil$	$O(n^{0.63})$	$\lceil \log n \rceil$	$O(n^{0.5})$	$O(n^{0.5})$	s
Upper Bound	$\lceil \frac{n+1}{2} \rceil$	$O(n^{0.63})$	$\lceil \frac{n+1}{2} \rceil$	$O(n^{0.5})$	$O(n^{0.5})$	$O(n)$

MQP: The Majority Quorum Protocol [12]. GP: The Grid Protocol [5].
HQP: The Hierarchical Quorum Protocol [8].HGP: The Hierarchical Grid Protocol [9].
TQP: The Tree Quorum Protocol [2].CP: The Column Protocol under $MC(k, s)$, where $k >> s$.

Table 1. Bounds on quorum sizes for various protocols.

and 4. When CP is concerned, we assume that replicas are arranged as $MC(5,3)$ in the 15-replica system, and as $MC(10) = (C_1, \ldots, C_{10})$, where $|C_1| = \cdots = |C_9| = 3$ and $|C_{10}| = 4$, in the 31-replica system (recall that we suggest adopting small column sizes except 2). The formulas for calculating the availabilities of ROWA, MQP and TQP are discussed below.

It is easy to see that ROWA's read and write availabilities are $1-(1-p)^n$ and p^n, respectively. MQP does not differentiate read quorums form write quorums. Its availability is given in [2] as
$Prob.(h$ replicas are up$) + Prob.(h+1$ replicas are up$) + \cdots + Prob.(n$ replicas are up$)$
$= \sum_{i=h}^{n} \binom{n}{i} p^i (1-p)^{n-i}$, where $h = \lceil \frac{n+1}{2} \rceil$.

Assuming data replicas are organized as a binary tree $\mathbf{T}$, the availability of TQP is given in [2] as
Availability($\mathbf{T}$) =
$Prob.(\mathbf{T}$'s root is up$) \times Availability(\mathbf{T}$'s left subtree$) \times Unavailability(\mathbf{T}$'s right subtree$) +$
$Prob.(\mathbf{T}$'s root is up$) \times Unavailability(\mathbf{T}$'s left subtree$) \times Availability(\mathbf{T}$'s right subtree$) +$
$Prob.(\mathbf{T}$'s root is up$) \times Availability(\mathbf{T}$'s left subtree$) \times Availability(\mathbf{T}$'s right subtree$) +$
$Prob.(\mathbf{T}$'s root is not up$) \times Availability(\mathbf{T}$'s left subtree$) \times Availability(\mathbf{T}$'s right subtree$)$.

Figures 3 and 4 reveals that CP has comparably high availability. The read availability and the write availability of ROWA are almost bounds of those of other protocols. The availability of TQP is better (respectively, worse) than that of MQP when up-probability is smaller (respectively, larger) than 0.5. For a wide range of up-probabilities, the read (respectively, write) availability of CP is a little better (respectively, worse) than the availability of TQP.

5. CONCLUSION

A quorum-based protocol, called column protocol, has been proposed in this paper for managing replicated data. This protocol is fault-tolerant; it allows replicated data to be accessed consistently even in the presence of network partitioning. By organizing data replicas into a multi-column structure, the protocol generates quorums of constant size in the best case. When some replicas are inaccessible, the quorum size increases gradually and may be as large as $O(n)$. Nevertheless, we have shown that the expected quorum size of the column protocol remains constant as n grows. This is a desirable property since the message cost for accessing replicated data is directly proportional to the quorum size. In addition, we have also shown that the availability of the column protocol is comparably high. The two properties—constant expected quorum size and comparably high availability—make the column protocol practical for managing replicated data.

Acknowledgements — I would like to thank the anonymous referees for their valuable comments, which helped for improving the paper. The close-form equations of availability and expected quorum size resulted from the comments of the referees.

REFERENCES

[1] D. Agrawal and A. El Abbadi. Exploiting logical structures in replicated databases. *Information Processing Letters*, **33**(5):255–260 (1990).

[2] D. Agrawal and A. El Abbadi. An efficient and fault-tolerant solution for distributed mutual exclusion. *ACM Transactions on Computer Systems*, **9**(1):1–20 (1991).

[3] P. A. Bernstein and N. Goodman. An algorithm for concurrency control and recovery in replicated distributed databases. *ACM Transactions on Database Systems*, **9**(4):596–615 (1984).

[4] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency control and recovery in database Systems.* Addison-Wesley (1987).

[5] S. Y. Cheung, M. H. Ammar, and M. Ahamad. The grid protocol: a high performance scheme for maintaining replicated data. *IEEE Transactions on Knowledge and Data Engineering*, **4**(6):582–592 (1992).

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

[7] J. A. Dossey, A. D. Otto, L. E. Spence, and C. V. Eynden. *Discrete Mathematics.* Scott, Foresman and Company (1986).

[8] A. Kumar. Hierarchical quorum consensus: a new algorithm for managing replicated data. *IEEE Transactions on Computers*, **40**(9):996–1004 (1991).

[9] A. Kumar and S. Y. Cheung. A high availability $\sqrt{n}$ hierarchical grid protocol for replicated data. *Information Processing Letters*, **40**(6):311–316 (1991).

[10] A. Kumar, M. Rabinovich, and R. K. Sinha. A performance study of general grid structures for replicated data. In *Proc. 13th Int. Conf. on Distributed Computing Systems*, pp. 178–185, Pittsburgh, PA (1993).

[11] M. L. Neilsen. *Quorum structures in distributed systems.* PhD thesis, Kansas State University (1992).

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