A Fully Distributed Quorum Consensus Method with High Fault-Tolerance and Low Communication Overhead*

Xuemin Lin
Department of Computer Science
The University of Western Australia
Nedlands, WA 6907, Australia
e-mail: lxue@cs.uwa.oz.au

Abstract

The main objective of data replication in a distributed database system is to provide high data availability for transaction processing. Quorum consensus (QC) methods are commonly applied to managing replicated data. In this paper, we present a new quorum consensus method. The proposed QC method is highly fault-tolerant, and fully distributed (i.e., each site in a distributed system is equally weighted). Further, we can show that the proposed QC method has a low message overhead: 1) In the best case, each transaction operation process needs only to communicate with $\Omega(\sqrt{n})$ remote sites to get permission (n is the number of sites storing replicated copies of the manipulating data item). 2) In the worst case, each transaction operation process may be forced to communicate with $\Omega(\sqrt{n}\log n)$ remote sites due to site failures. We also compare our method with the existing QC methods.

Keywords: Concurrency control, distributed computing, fault-tolerance, replicated data management, quorum consensus method.

1 Introduction

Distributed database system availability may be enhanced through data replication. However, mutual consistency among the replicated copies of data should be maintained by synchronizing transactions at different sites, so that a global serialization order can be

*The earlier version of this paper was published in CATS'96 ensured. Thus, an appropriate management of replicated data involves a compromise between two conflicting goals: maximizing data availability and maintaining consistency of data. *Quorum consensus* (QC) methods [5,6] are frequently used in managing replicated data.

Using a QC method, an operation of a transaction issued at a site can proceed only if permission is granted by a group of other sites storing the replicas of the manipulating data. A general protocol of a QC method for processing a transaction operation can be described as follows.

Given a data item (object), at each site s_i, the regulations for forming a read quorum group S_i^r and a write quorum group S_i^w are assigned, where S_i^r and S_i^w are both in terms of a subset of the sites storing the data object replicas, so that the intersection of a read quorum group and a write quorum group is not empty, neither is the intersection of two write quorum groups. These two *intersection invariants* can be formally defined as:

> for each pair of sites s_i and s_j (i and j may be the same), $S_i^w \cap S_j^w \neq \emptyset$ and $S_i^r \cap S_j^w \neq \emptyset$.

A read (write) operation should get permission from the *concurrency controller* at each site in S_i^r (S_i^w) before it is processed.

If a correct concurrency control mechanism [2] is applied, a QC method will enforce, through the intersection invariants, the situation that a write and a read cannot take place simultaneously on different copies of the same object, and similarly, neither can two writes. Thus, the *serializability* in concurrently executing transactions can be assured.

Recent trends in developing new QC methods include coupling high data availability with a low communication cost for processing transactions. The achievements of a low communication cost may be either

1. through the minimization of the number of remote sites with which a transaction operation process has to communicate. [7,8,11,12], or
2. through the minimization of the total communication cost for processing a given set of transactions [9,10].

In this paper, we restrict our interests to 1. Interested readers may refer to [9,10] for a detailed discussion about minimizing the total communication cost.

A number of QC methods have been proposed [1,4,3,7,8,11,12] to reduce quorum group sizes; and thus, the number of remote sites with which a transaction operation process has to communicate is reduced. Among them, the QC methods in [1,3] have the smallest quorum group size $\Omega(\log n)$, where n is the number of sites storing the manipulating data object. However, in these two approaches, each site is not equally weighted in performing a quorum group.

To maximize utilization of a distributed system, and then to enhance the overall performance of transaction processes, *fully distributed* QC approaches have been investigated [7,8,11,12]. In a fully distributed QC approach, each site is equally weighted in performing a quorum group. In this paper, we discuss only fully distributed QC methods.

It is shown [11] that in a fully distributed QC method, $\sqrt{n}$ is a lower bound of a quorum group size. This lower bound has been achieved by the QC method in [11]. A rigorous analysis [12] shows that by the QC method in [11], the "data availability" (to be defined in Section 2) for processing an individual operation gets smaller (falls asymptotically to 0) as the number of replicas of each data item is increased. This defeats the main objective of data replication, i.e., increasing data availability through replication. Thus, it is desirable that a QC method should guarantee the data availability asymptotically increasing to 1.

In [12], a fully distributed QC method is provided that guarantees the data availability asymptotically increasing to 1 as the number of replicas increases. However, each operation process is forced to communicate with $\Omega(\sqrt{n}\log^{0.5} n)$ remote sites.

Kumar and Cheung[8] proposed another fully distributed QC method, by which each operation process communicates with $\Omega(\sqrt{n})$ remote sites if no sites are down, but may be forced to communicate with $\Omega(n)$ in the worst case. In this paper, we will first prove that this QC method [8] has the property that the data availability is increasing (asymptotically to 1) as the number of replicas increases. Then, we present a novel QC method. The proposed QC method reduces the worst case message number in [8] from $\Omega(n)$ to $\Omega(\sqrt{n}\log n)$, while retains the other properties. Specifically, we can show that the proposed method is fully distributed, and guarantees that the data availability goes asymptotically to 1. Further,

- If at the time when an operation is being processed, there are no (or a few) sites with failures, then the operation process will communicate with only $\Omega(\sqrt{n})$ remote sites.

- If many sites have failures, an operation process may be forced to communicate with $\Omega(\sqrt{n}\log n)$ remote sites to confirm a quorum group.

Thus, the proposed QC algorithm is an improvement of the QC method in [8]. We will also show that the proposed QC method has a lower average message overhead than that in [7,12] in case where the site failure probabilities are low and n is large.

The rest of the paper is organized as follows. In Section 2, we first give our environment assumptions, and then introduce a mathematic model to justify the degree of fault-tolerance of a QC method. In Section 3, we present our QC method. Section 4 provides a rigorous performance analysis of our QC method, and a comparison between our QC method and the related fully distributed approaches. This is followed by a conclusion.

2 Preliminaries

In our distributed system environment, we assume that communication between different sites is through exchanging messages. Detection of a failure of a site by another site happens through sending a message, but receiving no reply. To simplify analysis of data availability, we assume that communication links never fail, and that each site failure probability is independent. The networks under consideration are *fully connected*; that is, a message can be sent directly between any pair of sites. We follow the model where replicated data is represented by multiple copies.

In this paper, we apply a simple transaction management model in which a given QC method is incorporated. We assume that each site has all necessary static information, such as the regulations for forming a quorum group and locations of data. Once a transaction is issued, it is first decomposed into *external* operations that access physically stored data, and *internal* operations that manipulate the retrived data. During transaction running time, the given QC method is invoked before processing an external operation besides an application of a concurrency control mechanism. Without loss of generality, we assume that an external operation is either a *simple* read or a *simple* write (that is, each read or write manipulates only one data object). Consequently, in this paper we need only to consider transactions that are either a simple read or a simple write. We study only a QC method with respect to one data object, Without loss of generality, we can also assume *full data replication*; that is, a copy of each object exists at all sites.

Suppose that in a distributed system N, the probability of each site being alive is given. Given a site s_i and a QC method A, let $R_{r,A,i}$ ($R_{w,A,i}$) be the probability of successfully confirming a read (write) quorum group at s_i by the QC method A. A QC method has *high site resilience* [12] if

$$\min_{s_i \in N, q \in \{r,w\}} R_{q,A,i} \to 1 \text{ when } n \to \infty,$$

where n is the number of sites in the network N. Note that a QC method with high site resilience can guarantee that in the asymptotical case, an alive site is always able to confirm a read (or write) quorum group for processing its issued transactions; that is, the data availability is increased through increasing the number of replicas, unlike the one in [11].

Note that we consider only the fault tolerance of transaction synchronization algorithms. So, as far as we are concerned, an operation process cannot successfully assemble a read (or write) quorum group *only* due to site failures. We do not consider the situation where an operation process fails because some other operation has already been granted permission to proceed.

3 A New QC approach

Inspired by the results in [12,8,7,11], we have developed a new QC approach on the top of a tree QC (TQC) approach and the grid QC method in [11,4]. In this section, we first review the method in [11]; and then present the TQC method. Finally, we will present our new approach. Although our TQC method can be equivalently transformed into the QC method in [8], the use of TQC in our new QC method can reduce the communication overhead in the worst case for TQC while retains the other advantages of TQC.

3.1 A Grid QC Method

A grid QC (GQC) method was independently proposed in [4,11], which is fully distributed. In [11], GQC is implemented as follows. The n sites are organized into a grid square, where possible dummy sites are placed at the grid positions in the upper right corner (see Figure 1(b)). To select a quorum group at site s_i, a random selection of a column and a row in the grid is made such that the selected row and column intersect at a non-dummy site. The sites in the selected column and row form a (read/write) quorum group S_i for s_i. (Figure 1 illustrates randomly selected quorum groups of site 4 and site 1 respectively for two different cases.) Clearly, each S_i has at most $2\lceil\sqrt{n}\,\rceil - 1$ sites, any two quorum groups have a non-empty intersection, and each site belongs to $\Omega(\sqrt{n})$ groups. If a failure of a site in S_i happens, then a transaction process using S_i has to wait until the failure recovers.

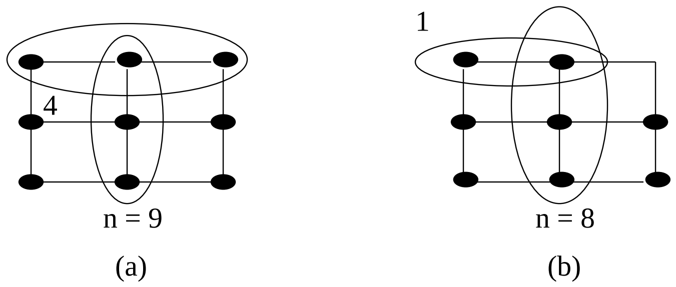

Figure 1: A grid method

In [4], a different implementation of GQC is introduced. Note that in GQC, there are n possible different quorum groups with respect to n sites. Each possible quorum group corresponds to a column and a row that intersect at a non-dummy site. Once a formed S_i fails, instead of waiting, a new quorum group will be elected. The experiments showed that when n is not very large, this algorithm gives a good data availability. However, there are two problems: 1) in the worst case, a site has to communicate with $\Omega(n)$ remote sites to confirm a quorum group, 2) the data availability still goes asymptotically to 0.

In this paper, we adopt the implementation by Maekawa [11]. Then, we show how we can improve its data availability by combining it with the tree QC approach.

3.2 A tree QC approach

Motivated by achieving the lower bound $\sqrt{n}$ in [11], we are interested in developing a distributed QC method with quorum group size $\Omega(\sqrt{n})$ such that it has high site resilience.

A rooted tree where each node has at most k children is called k-way tree. A *perfectly balanced* k-way tree has the property that every non-leaf node has exactly k-children. Consider a perfectly balanced k-way tree whose leaf set represents the set of sites in a distributed system, where k is an even number and the number of sites is denoted by n. Let us iteratively mark nodes from the root to the bottom as follows.

Mark Algorithm:

> We first mark the root. Then, iteratively a marked node marks half of its children.

Figure 2 shows one example, where marked nodes are dark coloured.

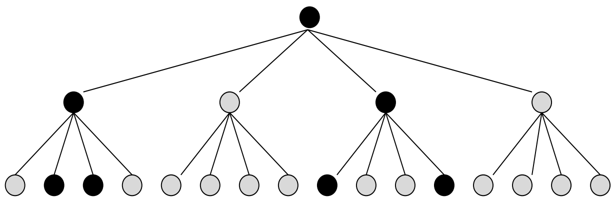

Figure 2: Mark Algorithm

By a simple calculation, we conclude that the number of marked leaves is $\frac{n}{2^{m-1}}$ where $n = k^{m-1}$ and m is the number of layers in the tree. It is clear that

- $\frac{n}{2^{m-1}} = \sqrt{n}$ if $k = 4$, and
- $\frac{n}{2^{m-1}} \geq n^{1+\log_6 2}$ if $k > 4$ and k is even.

Note that $1 + \log_6 2 > 0.5$. Suppose that the n sites in a given network can be represented as the leaf set of a perfectly balanced k-way tree, and we adopt the marked leaves as a "quorum" group. Clearly, $k = 4$ will give the smallest quorum group size $\Omega(\sqrt{n})$. This is the key of TQC. However, there are two problem we have to solve for using this idea in the development of TQC:

1. In most cases, n cannot be represented as the cardinality of the leaf set of a perfectly balanced 4-way tree. How do we find an alternative 4-way tree such that the size of marked leaves is about $\sqrt{n}$?
2. The above marked leaves, actually, cannot be used as a quorum group, since two different groups of randomly marked leaves do not necessarily have an intersection.

The first problem is easy to solve; and our method is to build an appropriate balanced tree. Suppose that in a distributed system, there are n sites. Let $m(n)$ denote $\lfloor \log_4 n \rfloor$. Clearly,

- $4^{m(n)} \leq n < 4^{m(n)+1}$, and
- $4^{m(n)} = n$ if and only if $m(n) = \log_4 n$.

Given n, a perfectly balanced 4-way tree with $m(n)+1$ layers is denoted by $T_{4,n}$. Obviously, if $\log_4 n$ is an integer then the n sites can be represented as the leaf set of $T_{4,n}$.

In case where $m(n) \neq \log_4 n$ (i.e., $n > 4^{m(n)}$), the number $4^{m(n)}$ of leaves in $T_{4,n}$ is smaller than n. Thus, after mapping each leaf in $T_{4,n}$ to a different site, the number of remaining (unmapped) sites is $n - 4^{m(n)}$ and is denoted by $R(n)$. Note that $R(n) < 3 \times 4^{m(n)}$. It implies that we need only to extend $T_{4,n}$ by one more layer, to accommodate the n sites by the leaves. The extension is described as follows.

Let $c(n) \triangleq R(n) \bmod 3$, and $I(n) \triangleq \frac{R(n)-c(n)}{3}$. It is clear that $c(n) \in \{0, 1, 2\}$, $I(n)$ is an integer, and $R(n) = 3I(n) + c(n)$. To extend $T_{4,n}$, we first randomly choose either

- $I(n) + 1$ leaves from $T_{4,n}$ if $c(n) \neq 0$, or
- $I(n)$ leaves from $T_{4,n}$ if $c(n) = 0$.

Then, for each leaf v in the first chosen $I(n)$ leaves of $T_{4,n}$, four new leaves are attached as the children of v, while the $(I(n)+1)$th chosen leaf of $T_{4,n}$ will be attached, as its children, $c(n) + 1$ leaves. The obtained tree is called a TQC tree, and is denoted by T_n. It can be immediately verified that the size of the leaf set of T_n is n, and thus the leaf set can accommodate the n sites. Figure 3 illustrates all possible cases ($n = 7, 8, 9, 16$), where leaves are dark coloured.

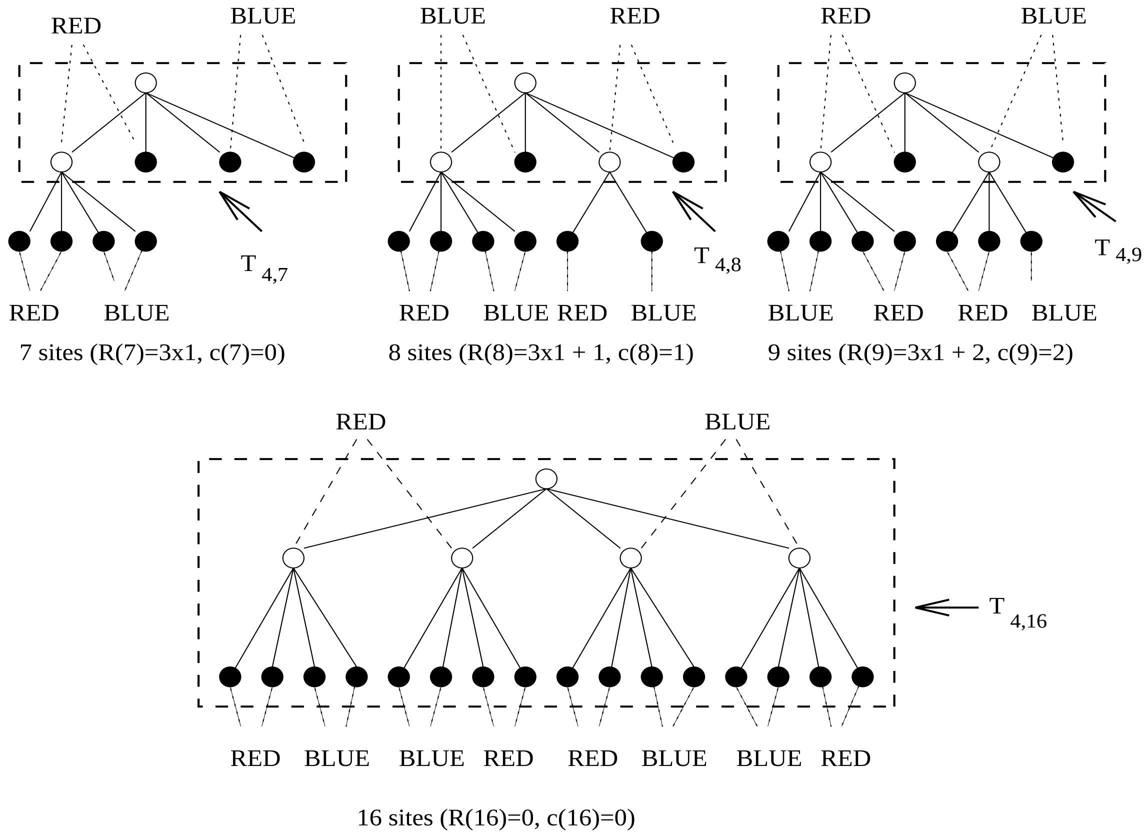

Figure 3: TQC Trees

Our approach to the second problem is to set up two classes of groups of marked leaves, such that each group in one class will definitely have an intersection with each group in another class. A quorum group will be formed as the union of two groups respectively from the two classes. This will guarantee the two intersection invariants. In the meantime, the size of a quorum group remains $\Omega(\sqrt{n})$. Our approach is detailed as follows.

Given n and its TQC tree T_n, we evenly divide the children of each internal node v into two disjoint groups: Red and Blue.

- If v has 4 children, then we put 2 children in Red group, and the other 2 children in Blue group.
- If v has 3 children, the we put 2 children in Red group, and the other 1 child in Blue group.
- If v has 2 children, then we put one in Red group, and another in Blue group.

See Figure 3 for example.

Once the nodes have been coloured, we can set up the regulations for producing two different classes of groups Q_s and Q_c of marked leaves.

- Forming a Q_c: In Mark Algorithm, each marked node randomly marks its two **cross** coloured children. The marked leaves form a Q_c.
- Forming a Q_s: In Mark Algorithm, a marked node first randomly chooses a colour, and then makes its **same** coloured children with respect to the chosen colour. The marked leaves form a Q_s.

The union of Q_c and Q_s forms a (read/write) quorum group. Note that at each site, the regulations for performing a read quorum group and a write quorum group is the same - using the union of Q_c and Q_s.

A TQC method can be precisely described as follows. Assume that T_n are given a *consecutive* layering: 1) the layer number of the root is 0, and 2) the layer number of a parent is less than that of its children by one. We use L_i to denote the set of nodes with layer number i, and is called ith layer.

TQC Method:

Phase 1: A corresponding TQC tree T_n is built, and then the nodes in T_n are evenly coloured. Each site keeps the information of T_n and the nodes colouring. Note that this is a static phase which should be done before processing any transaction.

Phase 2: At running time of a transaction, both Q_c and Q_s are iteratively assembled from L_0 to the bottom layer, as follows. The union of formed Q_s and Q_c is output as a selected quorum group for processing the transaction.

(2.a) To form a Q_c, the root randomly marks one of its children in the red group, and one of its children in the blue group. Iteratively, each marked node at L_i, except a leaf, marks one of its children in the red group, and one of its children in the blue group. Q_c is formed by collecting the marked leaves.

(2.b) To form a Q_s, the root randomly chooses a colour (red or blue), and then marks its children in the chosen colour group. Iteratively, each marked node, except a leaf, randomly chooses a colour (red or blue), and then marks its children in the chosen colour group. Q_s is the set of the marked leaves.

In case that some marked leaves (sites) in Q_s are not available due to their site failures, we need to perform Q_s again. Note that the detection of the failures of a site by another site is assumed through sending a message. In order to save communication costs among remote sites, we should do "minimal modification" on failed Q_s to get a new Q_s:

> Suppose that a marked site l in Q_s, say l in red group, has a failure. The transaction issuing site j looks up the TQC tree to find the parent $pa1$ of site l. Then re-do (2.b) from $pa1$ to mark all its children in blue group. After this, if we find that it is impossible to assemble Q_s with respect to $pa1$ (i.e., one of its children in blue group has a failure), then j looks up the TQC tree again to get the parent $pa2$ of $pa1$, say $pa1$ in blue group. Then re-do (2.b) from $pa2$ to mark its children in red group, and so on. Finally, we can determine whether or not it is possible to assemble a Q_s with alive sites.

Similarly, in case that some sites in a formed Q_c have failures, we can modify Q_c into a new Q_c from the bottom of the TQC tree:

> Suppose that a marked site l in Q_c has a failure. The transaction issuing site j looks up the TQC tree to find the parent $pa1$ of site l. Then re-do (2.a) from $pa1$: if another originally marked child, in Q_c, of $pa1$ is alive, then we keep it being marked in the new forming Q_c. After this, if we find that it is impossible to assemble Q_c with respect to $pa1$, then j looks up the TQC tree again to get the parent $pa2$ of $pa1$. Then re-do (2.a) from $pa2$, where if another originally marked child, in Q_c, of $pa2$ is alive, then we keep it being marked in the new forming Q_c; and so on. Finally, we can determine whether or not it is possible to assemble a Q_c with alive sites.

For example, consider Figure 4. It illustrates a set of 16 sites organized into a three layer TQC tree. The variously possible Q_c can be: $\{1,3,14,15\}$, $\{5,8,14,16\}$, etc. Meanwhile, the possible Q_s may be $\{3,4,5,6\}$, $\{9,10,13,14\}$, etc. Hence, a quorum group (the union of a formed Q_c and a formed Q_s) may be $\{1,3,4,5,6,14,15\}$, $\{3,4,5,6,8,14,16\}$, $\{1,3,9,10,13,14,15\}$, $\{5,8,9,10,13,14,16\}$, etc. If $Q_c = \{1,3,14,15\}$ is formed, then we find that site 15 is not available. The TQC method will either form the new $Q_c = \{1,3,14,16\}$ if site 14 does not have a failure, or form the new $Q_c = \{1,3,13,16\}$ if both site 14 and site 15 have failures. Suppose that in the new formed $Q_c = \{1,3,14,16\}$, site 16 also has a failure. Then one of $\{9,11\}$, $\{9,12\}$, $\{10,11\}$ and $\{10,12\}$ will be used to replace $\{14,16\}$.

Similarly, if $Q_s = \{3,4,5,6\}$ is formed, then we find that the site 6 is not available. The new formed Q_s will be $\{3,4,7,8\}$, and so on.

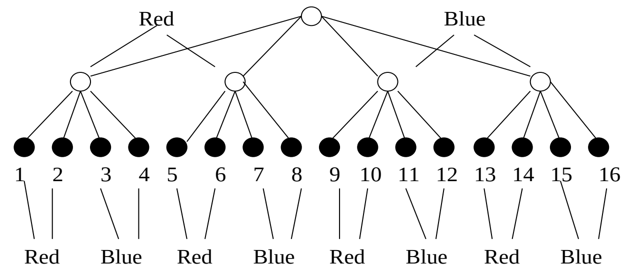

Figure 4: TQC Method

Clearly, TQC is fully distributed. It can be immediately verified that in TQC, any pair of Q_c and Q_s have one common site. Thus, the TQC method is *correct* (i.e., each pair of quorum groups have at least one common site), since a quorum group in TQC is the union of a Q_c and a Q_s.

In the TQC method, the maximal size of a Q_c is:

$$2^{m(n)+1} \leq 2^{\log_4 n+1} = 2\sqrt{n}.$$

Similarly, the maximum size of Q_s is bounded by (not greater than) $2\sqrt{n}$. So, the size of a quorum group is bounded by (not greater than) $4\sqrt{n}-1$ (noting a Q_c and a Q_s has a common site). This means that an operation process needs to communicate with only $\Omega(\sqrt{n})$ remote sites to confirm (a read or write) quorum group, provided that there is no site with a failure while processing an operation.

Suppose that at the time when an operation is issued, some sites are not available due to their failures. An operation process, using TQC, may be forced to communicate with more than half of the sites in a network to confirm whether or not it can successfully assemble a (read or write) quorum group. Meanwhile, in TQC, each remote site is communicated with at most once in forming a quorum group. Thus, in the worst case, a quorum group is formed by communicating with $\Omega(n)$ remote sites.

3.3 Our New QC method - TMQC

In this sub-section, we present our quorum consensus approach - TMQC method, which logically nests TQC and GQC (by Maekawa) together. In the TMQC method, the regulations of forming read and write groups are the same at each site. We first group n sites $\{s_i : 1 \leq i \leq n\}$ into k disjoint groups $\{G_i : 1 \leq i \leq k\}$ such that these groups have an almost equal size. A quorum group will be formed, in TMQC, through two layered construction. TMQC first performs GQC method on these k groups to obtain a "quorum group" A_j - a subset of $\{G_i : 1 \leq i \leq k\}$. By the application of TQC to each element G_i in A_j, we get a quorum group Q_i corresponding to each G_i in A_j; the union of all these Q_i forms a quorum group in TMQC. TMQC can be precisely described as follows.

TMQC Method:

Step 1: Group the n sites into k disjoint subgroups $\{G_i : 1 \leq i \leq k\}$ such that

$$\lfloor \frac{n}{k} \rfloor \leq |G_i| \leq \lceil \frac{n}{k} \rceil, \tag{1}$$

and such that $\cup_{i=1}^{k} G_i = \{s_i : 1 \leq i \leq n\}$. A mapping g_1, from $\{s_i : 1 \leq i \leq n\}$ to $\{G_i : 1 \leq i \leq k\}$, is constructed such that $g_1(s_i) = G_j$ if $s_i \in G_j$. Go to Step 2.

Step 2: View each element G_i in $\{G_i : 1 \leq i \leq k\}$ as a "site". Applying GQC to these k "sites" $\{G_i : 1 \leq i \leq k\}$, there are k possible "quorum groups" $\{A_i : 1 \leq i \leq k\}$. Each A_i consists of the column and row intersecting at G_i in the formed grid square, and A_i is a subset of $\{G_i : 1 \leq i \leq k\}$. Using GQC to form a "quorum group" at "site" G_i, a random selection of an A_j from $\{A_i : 1 \leq i \leq k\}$ is made; and we denote the random selection by $g_2(G_i)$ where g_2 is a random function and returns an element in $\{A_i : 1 \leq i \leq k\}$. Note that $1 \leq i \leq k$, $|A_i| \leq 2\lceil\sqrt{k}\,\rceil$. Go to Step 3.

Step 3: For each site s_i, its (read or write) quorum group is formed, which consists of certain sites in group $g_2(g_1(s_i))$ (say A_j), such that for each element G_x in A_j, we use TQC to form a (read or write) quorum group Q_x^i in G_x. Then the quorum group of s_i is $\cup_{G_x \in g_2(g_1(s_i))} Q_x^i$.

The TMQC method is a fully distributed method, since it is a combination of the fully distributed methods - TQC and GQC. It can be immediately verified that TMQC is also correct (i.e, each two quorum groups have at least one common site), based on the facts that both of the methods - TQC and GQC, are correct.

Figure 5 illustrates an example of how TMQC works. There are 36 sites. Let $k = 4$, $G_1 = \{s_i : 1 \leq i \leq 9\}$, $G_2 = \{s_i : 10 \leq i \leq 18\}$, $G_3 = \{s_i : 19 \leq i \leq 27\}$, $G_4 = \{s_i : 28 \leq i \leq 36\}$. (Note that $g_1(s_{9i+k}) = G_{i+1}$ for $0 \leq i \leq 3$ and $1 \leq k \leq 9$.) By an application of GQC to G_i for $1 \leq i \leq 4$, we assume the grid square is organized as illustrated in Figure 5 (connected by the thick lines); and the 4 possible "quorum groups" on "sites" G_is are $A_1 = \{G_1, G_2, G_3\}$, $A_2 = \{G_1, G_2, G_4\}$, $A_3 = \{G_1, G_3, G_4\}$, $A_4 = \{G_2, G_3, G_4\}$. Suppose that a quorum group at site s_1 is required, and the random function g_2 returns the value A_4 for $g_2(G_1)$ at this time. Thus, a quorum group at s_1 will consist of certain sites in $g_2(g_1(s_{28}))) = A_4$, that is, certain sites in G_2, G_3, and G_4. By applying TQC to each of G_2, G_3, and G_4, the possible quorum groups at s_1 will be $\{s_{10}, s_{11}, s_{12}, s_{14}, s_{15}, s_{18}, s_{24}, s_{25}, s_{26}, s_{27}, s_{28}, s_{29}, s_{30}, s_{34}, s_{35}\}$, etc.

Now, we estimate the number of remote sites which will be accessed by using TMQC in a transaction process. Clearly, if there are no sites with a failure, in TMQC a transaction process will communicate with at most $(2\lceil\sqrt{k}\,\rceil)(4\sqrt{\frac{n}{k}} - 1)$ remote sites.

$$(2\lceil\sqrt{k}\,\rceil)(4\sqrt{\frac{n}{k}} - 1) = \Omega(\sqrt{n}). \tag{2}$$

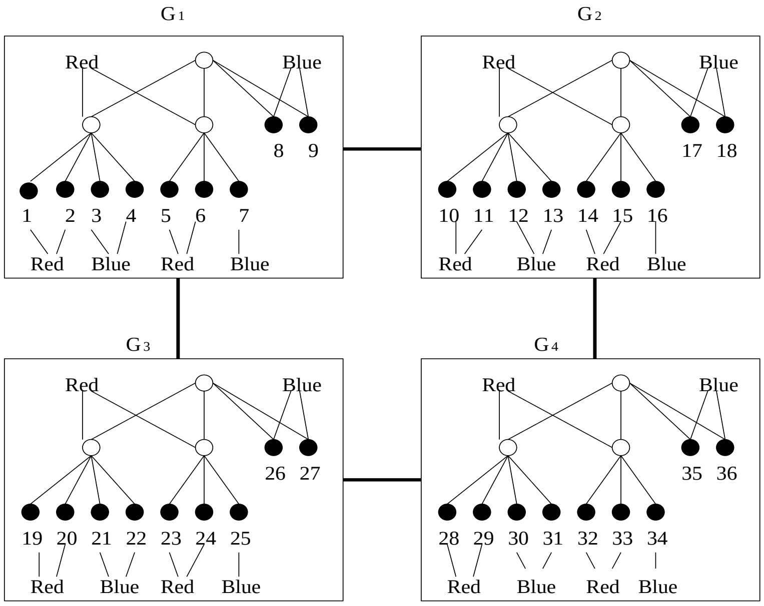

Figure 5: TMQC approach

Note that the constant associated with $\sqrt{n}$ in the above $\Omega(\sqrt{n})$ is 8 which is twice of the constant 4 associated with $\sqrt{n}$ in TQC. In fact, a QC method in [11] other than GQC has quorum size $\sqrt{n}$. If we apply that QC method [11] to TMQC, then the constant associated with $\sqrt{n}$ will remain 4 in TMQC; and the other results related to TMQC proven in the next section will also hold. Since that QC method [11] is complicated to describe, in this paper we apply GQC for the simplicity to illustrate TMQC.

If many sites are not available in the network, then to perform a quorum group Q_x^i in TMQC with respect to each relevant G_x, $\Omega(|G_x|)$ remote sites may be communicated with. So in the worst case, a transaction process by using TMQC will communicate with the following number of remote sites.

$$\Omega(\lceil \sqrt{k} \rceil \lceil \frac{n}{k} \rceil) = \Omega(\frac{n}{\sqrt{k}}). \tag{3}$$

4 Performance Analysis of TMQC

We use n to denote the number of sites in a distributed system. In this section, we show that in TMQC, an appropriate choice of k can lead to high site resilience, and (3) will be $\Omega(\sqrt{n}\log n)$.

To analyze the data availability of TMQC, we first analyze TQC. In TMQC, with respect to each selected subgroup G_i, we use $p1(G_i)$ to denote the probability of failing to form any Q_c in G_i by TQC, while we use $p2(G_i)$ to denote the probability of failing to form any Q_s in G_i by TQC.

Theorem 1 *Suppose that in a distributed system, the probability of each site failure is q, and $4q < 1$. In a TMQC method, let the size of a selected subgroup G_j be l. Then $p1(G_j) \leq \frac{(2f(q))^{\frac{1}{2}l^{\frac{1}{2}}}}{2}$ where $f(q) = 2q - q^2$, and $p2(G_j) \leq \frac{(4q)^{\frac{1}{2}l^{\frac{1}{2}}}}{4}$.*

Proof: Note that in TQC, G_j has been represented as the leaf set of a TQC tree, as described in the last section. It can be immediately verified that the tree has $m(l) + 2$ layers if $m(l) \neq \log_4 l$, otherwise the tree has $m(l) + 1$ layers. Suppose that the maximum failure probability to form a Q_c in a sub-rooted tree of G_i with its root at L_i ($0 \leq i \leq m(l)$) is q_{1,L_i}. Assume that the maximum failure probability to form a Q_s in a sub-rooted tree with the root at L_i ($0 \leq i \leq m(l)$) is q_{2,L_i}.

Note that each node at layer $L_{m(l)}$ has either: 4 children, or 3children, or 2 children, or no child.

Thus, the failure probability for forming a Q_c in a sub-rooted tree with its root at $L_{m(l)}$ is either: $1 - (1 - q^2)(1 - q^2)$ which is not greater than $f(q)$, or $1 - (1 - q)(1 - q^2)$ which is not greater than $f(q)$, or $f(q)$, or q which is not greater than $f(q)$.

Hence, we have $q_{1,L_{m(l)}} \leq f(q)$. Meanwhile, the failure probability for forming any Q_s from each sub-rooted tree with its root at $L_{m(l)}$ is either: $(1 - (1 - q)^2)^2$ which is not greater than q when $4q < 1$, or $(1 - (1 - q)^2)q$ which is not greater than q, or q^2 which is not greater than q, or q.

Therefore, $q_{2,L_{m(l)}} \leq q$. Below, we will prove that for $0 \leq i \leq m(l) - 1$,

$$q_{1,L_i} \leq 2q^2_{1,L_{i+1}}, \tag{4}$$

$$q_{2,L_i} \leq 4q^2_{2,L_{i+1}}. \tag{5}$$

Without loss of generality, we assume $n > 4$. Suppose that t is a sub-rooted tree at L_i for $0 \leq i \leq m(l) - 1$. According to the construction of a TQC tree, the root of t has 4 children. Let t_1, t_2, t_3 and t_4 be the different sub-rooted trees whose roots are the children of t, and at L_{i+1}. Without loss of generality, we assume that the roots of t_1 and t_2 are in the red group, and the roots of t_3 and t_4 are in the blue group. Furthermore, $q11$, $q12$, $q13$, $q14$ and $q1(t)$ are respectively the failure probabilities of forming any Q_c in t_1, t_2, t_3 and t_4 and t. Similarly, $q21$, $q22$, $q23$, $q24$ and $q2(t)$ are respectively the failure probabilities of forming any Q_s in t_1, t_2, t_3, t_4 and t. From TQC, it follows that $q1(t)$ is equal to

$$1 - (1 - q11q12)(1 - q13q14) = q11q12 + q13q14 - q11q12q13q14. \tag{6}$$

For $0 \leq q11, q12, q13, q14 \leq 1$, one may verify that (6) is a monotonic increasing function with respect to each variable $q1j$ for $1 \leq j \leq 4$. Note that for $1 \leq j \leq 4$, $q1j \leq q_{1,L_{i+1}}$. This implies that $q1(t) \leq 2q^2_{1,L_{i+1}} - q^3_{1,L_{i+1}}$. Hence, (4) holds.

Also, it follows from TQC that $q2(t)$ is equal to

$$(1 - (1 - q21)(1 - q22))(1 - (1 - q23)(1 - q24)).$$

Similarly, it follows that $q2(t) \le 4q_{2,L_{i+2}}^2 - 4q_{2,L_{i+2}}^3 + q_{2,L_{i+2}}^4$. Thus (5) holds.

From (4), it follows that

$$q_{1,L_0} \le 2^{\sum_{i=0}^{m(l)-1} 2^i} \times q_{1,L_{m(l)}}^{2^{m(l)}}. \qquad (7)$$

(7) can be written as

$$q_{1,L_0} \le 2^{2^{m(l)}-1} \times q_{1,L_{m(l)}}^{2^{m(l)}}.$$

So, $q_{1,L_0} \le \frac{(2q_{1,L_{m(l)}})^{2^{m(l)}}}{2}$.

Since $q_{1,L_{m(l)}} \le f(q)$, we have $q_{1,L_0} \le \frac{(2f(q))^{2^{m(l)}}}{2}$. From the facts that

- $m(l) = \lfloor \log_4 l \rfloor \ge \log_4 l - 1$,
- $2f(q) < 1$, and
- $q_{1,L_0} = p1(G_j)$,

it follows that $p1(G_j) \le \frac{(2f(q))^{\frac{1}{2}2^{\log_4 l}}}{2}$.

Note that $2^{\log_4 l} = l^{\log_4 2} = l^{\frac{1}{2}}$. It implies that the $p1(G_j) \le \frac{(2f(q))^{\frac{1}{2}l^{\frac{1}{2}}}}{2}$. Similarly, by noting $4q < 1$ and $q_{2,L_0} = p2(G_j)$, we can prove that $p2(G_j) \le \frac{(4q)^{\frac{1}{2}l^{\frac{1}{2}}}}{4}$. □

From the proof of Theorem 1, it follows that the restriction to an equal failure probability at each site is not necessary. So we have the following Corollary.

Corollary 1 *Suppose that in a distributed system, the maximum value of each site failure probability is p, and $4p < 1$. In TMQC, let the size of a selected subgroup G_j be l. Then $p1(G_j) \le \frac{(2f(p))^{\frac{1}{2}l^{\frac{1}{2}}}}{2}$ where $f(p) = 2p - p^2$, and $p2(G_j) \le \frac{(4p)^{\frac{1}{2}l^{\frac{1}{2}}}}{4}$.*

Let $p(G_j)$ denote the failure probability of forming a (read or write) quorum group in a G_j by TQC. In TQC, a quorum group of G_j is formed as the union of a Q_c and a Q_s. Thus $p(G_j) \le p1(G_j) + p2(G_j)$. Hence:

Corollary 2 *Suppose that in a distributed system, the maximum value of each site failure probability is p, and $4p < 1$. In TMQC, let the size of a selected subgroup G_j be l. Then $p(G_j) \le \frac{(2f(p))^{\frac{1}{2}l^{\frac{1}{2}}}}{2} + \frac{(4p)^{\frac{1}{2}l^{\frac{1}{2}}}}{4}$ where $f(p) = 2p - p^2$.*

Next, we show that in TMQC, a proper choice of k may lead to high site resilience. In a TMQC method, let b denote the minimum size of these subgroups G_i, that is, $b = \min_{1 \le i \le k}\{|G_i|\}$. From (1), it follows that $b = \lfloor \frac{n}{k} \rfloor \le \frac{n}{k}$. So,

$$k \le \frac{n}{b}. \qquad (8)$$

Theorem 2 *In a distributed system N with n sites, suppose that p is the maximum value of each site failure probability, and $0 \leq p < 0.25$. Further, in a TMQC method, $b = (C \log_2 n)^2$ where C is a constant, and $C \geq \frac{1}{\log_2 \frac{1}{4p}}$. Then $\min_{s_i \in N, q \in \{r,w\}} R_{q,TMQC,i} \rightarrow 1$ as $n \rightarrow \infty$.*

Proof: Note that in TMQC, the regulations for performing a read quorum group and a write quorum group are the same at each site. Thus, $R_{r,TMQC,i} = R_{w,TMQC,i}$ for each site s_i. This means we need only to prove that $\min_{s_i \in N} R_{w,TMQC,i} \rightarrow 1$ as $n \rightarrow \infty$.

Given a site s_m, its quorum group should be formed from certain sites of the group A_j $(= g_2(g_1(s_m)))$ according to the TMQC method. Let l_x denote the size of an element G_x in A_j. Clearly,

$$R_{w,TMQC,m} = \prod_{G_x \in A_j} (1 - p(G_x)).$$

Note $0 \leq p < 0.25$. So we can apply Corollary 2. From Corollary 2, it follows that,

$$R_{w,TMQC,m} \geq \prod_{G_x \in A_j} (1 - \frac{(2f(p))^{\frac{1}{2} l_x^{\frac{1}{2}}}}{2} - \frac{(4p)^{\frac{1}{2} l_x^{\frac{1}{2}}}}{4}). \tag{9}$$

Note that $|A_j| \leq 2\lceil \sqrt{k}\ \rceil \leq 2\sqrt{k} + 2$, $l_x \geq b$, and $0 \leq 1 - \frac{(2f(p))^{\frac{1}{2} l_x^{\frac{1}{2}}}}{2} - \frac{(4p)^{\frac{1}{2} l_x^{\frac{1}{2}}}}{4} \leq 1$ for each l_x. Let $P = \frac{(2f(p))^{\frac{1}{2} b^{\frac{1}{2}}}}{2} + \frac{(4p)^{\frac{1}{2} b^{\frac{1}{2}}}}{4}$. We can immediately verify that

$$R_{w,TMQC,m} \geq (1 - P)^{2\sqrt{k}+2}. \tag{10}$$

It is easy to verify that for any positive z such that $z \geq 1$, and for any non-negative y such that $1 \geq y$, we have:

$$z(1 - y) \geq 1 - y^z. \tag{11}$$

It immediately implies that $(2\sqrt{k} + 2)(1 - (1 - P)) \geq 1 - (1 - P)^{2(\sqrt{k}+1)}$. So,

$$R_{w,TMQC,m} \geq 1 - 2(\sqrt{k} + 1)P.$$

From (8), it follows that:

$$R_{w,TMQC,m} \geq 1 - 2(\sqrt{\frac{n}{b}} + 1)\ P. \tag{12}$$

(12) can be written as:

$$R_{w,TMQC,m} \geq 1 - 2(\sqrt{\frac{n}{b}} + 1)(\frac{(2f(p))^{\frac{1}{2}b^{\frac{1}{2}}}}{2} + \frac{(4p)^{\frac{1}{2}b^{\frac{1}{2}}}}{4}). \quad (13)$$

Note $2f(p) \leq 4p$. From (13), it follows that $R_{w,TMQC,m} \geq 1 - 2(\sqrt{\frac{n}{b}} + 1)\frac{3(4p)^{\frac{1}{2}b^{\frac{1}{2}}}}{4}$. Note $(4p)^{\frac{1}{2}} = 2^{\log_2 \sqrt{4p}}$ and $b = (C \log_2 n)^2$. Thus, we have that

$$R_{w,TMQC,m} \geq 1 - \frac{3}{2}(\sqrt{\frac{n}{b}} + 1)\ 2^{(\ \log_2 \sqrt{4p}\ ) \times C \log_2 n}.$$

It can be re-written as:

$$R_{w,TMQC,m} \geq 1 - \frac{3}{2\sqrt{b}}\ n^{C \log_2 \sqrt{4p}\ +\ 0.5} \ - \ \frac{3}{2}\ n^{C \log_2 \sqrt{4p}}. \quad (14)$$

Any C, such that $C \geq \frac{1}{log_2 \frac{1}{4p}}$, can be expressed as $C = \frac{1}{log_2 \frac{1}{4p}} + a$ where $a \geq 0$. So, (14) can be re-written as:

$$R_{w,TMQC,m} \geq 1 - \frac{3}{2\sqrt{b}}\ n^{0.5\ -\ \frac{1}{log_2 \frac{1}{4p}} \log_2 \frac{1}{\sqrt{4p}}\ -\ a \log_2 \frac{1}{\sqrt{4p}}} - \frac{3}{2} n^{-\ \frac{1}{log_2 \frac{1}{4p}} \log_2 \frac{1}{\sqrt{4p}}\ -\ a \log_2 \frac{1}{\sqrt{4p}}}.$$

Again, it can be re-written as:

$$R_{w,TMQC,m} \geq 1 - \frac{3}{2\sqrt{b}}\ n^{-\ a \log_2 \frac{1}{\sqrt{4p}}} \ - \ \frac{3}{2}\ n^{-0.5 - a \log_2 \frac{1}{\sqrt{4p}}}.$$

Note that $\frac{1}{\sqrt{4p}} > 1$. This leads to $\log_2 \frac{1}{\sqrt{4p}} > 0$.

Since $a \geq 0$, and $b \to \infty$ as $n \to$, we have $\min_{s_m \in N} R_{w,TMQC,m} \to 1$ as $n \to \infty$, no matter whether or not $a = 0$. □

Theorem 2 says:

> In a class π of distributed systems where the maximum site failure probability through the whole class is p and $1 \leq p \leq 0.25$, the TMQC method has *high resilience* if with respect to each distributed system with n sites in π, b is always chosen as $(C \log_2 n)^2$ where n is the number of sites in a distributed system, C is a constant over π, and $C \geq \frac{1}{\log_2 \frac{1}{4p}}$.

The choices of C and b together with (8) immediately imply that if with respect to any distributed system with n sites in π, k is always chosen as (noting (8)):

$$\max\{1,\ \min\{\lfloor \frac{n}{C^2 \log_2^2 n} \rfloor, n\}\}, \quad (15)$$

then the TMQC method has high site resilience. From (2), (3), and (15), it immediately follows that besides high site resilience over π, TMQC has the following properties,

- If there are no sites with a failure, a operation process will communicate with $\Omega(\sqrt{n})$ remote sites. (If only a few sites have a failure, this property still holds.)
- If a large number of sites have a failure, in the worst case an operation process will be forced to communicate with $\Omega(\sqrt{n}\log n)$ remote sites.

Now, we make a comparison between our TMQC method and those fully distributed QC methods in [7,11,12]. To simplify the description, we use HQC to denote the method in [7], GQC to denote the method in [11], RST to denote the method in [12].

First, TMQC, HQC, and RST all have high site resilience if the failure probability of each site is smaller than 0.25%, but GQC does not have. According to this index, TMQC, HQC, and RST are equally ranked first, while GQC is ranked fourth.

Now we compare the numbers of remote sites needed to be communicated in HQC, HMV, GQC, and TMQC.

In HQC, an operation process needs to communicate with $\Omega(n^{0.63})$ remote sites in the best case, and $\Omega(n)$ in the worst case. In RST, an operation process needs to communicate with $\Omega(\sqrt{n}\log^{0.5} n)$ remote sites in both the best case and the worst case. In TMQC, an operation process needs to communicate with $\Omega(\sqrt{n})$ remote sites in the best case, and $\Omega(\sqrt{n}\log n)$ remote sites in the worst case. In GQC, an operation process needs only to communicate with $\Omega(\sqrt{n})$ remote sites in both the best case and worst case.

According to this index, GQC is ranked first and HQC is ranked fourth. A further comparison between RST and TMQC must be carried out according to the average numbers of remote sites to be communicated for forming a quorum group. Our implementation results showed that the average numbers of remote sites to be communicated are also determined by the site failure probabilities. In fact, it can be shown that if site failure probabilities are low and n is large, then the average number of communicated sites using TMQC is less than that using RST. A proof is outlined as follows.

In [13], it showed that the average message number for each QC method can be modelled as a function which is polynomial with respect to site failure probabilities. This implies that the average message numbers can be described as a continuous function with respect to the site failure probabilities. Note that if each site failure probability is zero, then the average message number for TMQC is certainly $\Omega(\sqrt{n})$, while the average message number for RST is $\Omega(\sqrt{n}\log^{0.5} n)$. If n is large, then $\Omega(\sqrt{n}) < \Omega(\sqrt{n}\log^{0.5} n)$. Thus, the average message number for TMQC is lower than that for RST if site failure probabilities are low (say, very close to 0) and n is large.

Based on the above comparison, TMQC should be the best choice in case if site failure probabilities are low and n is large.

5 Conclusion

In this paper, we presented a novel fully distributed quorum consensus method, TMQC. TMQC has high site resilience, and a low message overhead. We also presented a comparison between our results and the other results.

References

[1] D. Agrawal and A. El Abbadi, An Efficient Solution to the Distributed Mutual Exclusion Problem, *ACM Transactions on Computer Systems*, 9(1), 1-20, 1991.

[2] P. Bernstein, V. Hadzilocs and N. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, Reading, Mass., 1987.

[3] H. Chen and J. Tang, An Efficient Method for Mutual Exclusion in Truly Distributed Systems, *International Conference on Distributed Computing Systems*, 97-104, 1994.

[4] S. Y. Cheung, M. 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.

[5] S. B. Davidson, H. Garcia-Molina and D. Skeen, Consistency in Partioned Networks, *ACM Computing Surveys*, 17(3), 341-370, 1985.

[6] H. Garcia-Molina and D. Barbara, How to Assign Votes in a Distributed Systems, *J. ACM*, 32(4), 841-860, 1985.

[7] A. Kumar, Hierarchical Quorum Consensus: A New Algorithm for Managing Replicated Data, *IEEE Transactions on Computers*, 40(9), 996-1004, 1991.

[8] A. Kumar and S. Y. Cheung, A $\sqrt{n}$ High Availability Hierarchical Grid Algorithm for Replicated Data, *Information Processing Letters*, 40, 311-316, 1991.

[9] A. Kumar and A. Segev, Cost and Availability Tradeoffs in Replicated Data Concurrency Control, *ACM Transactions on Database Systems*, 18(1), 102-131, 1993.

[10] X. Lin and M. E. Orlowska, An Optimal Voting Schema for Minimizing the Overall Communication Cost in Replicated Data Management, *Journal of Parallel and Distributed Computing*, 35, 97-103, 1996.

[11] M. Maekawa, A $\sqrt{N}$ Algorithm for Mutual Exclusion in Decentralized Systems, *ACM Transactions on Computer Systems*, 3(2), 145-159, 1985.

[12] S. Rangarajan, S. Setia and S. K. Tripathi, A Fault-tolerant Algorithm for Replicated Data Management, *IEEE Proceedings of the 8th International Conference on Data Engineering*, 230-237, 1992.

[13] S. Rangarajan, S. Setia and S. K. Tripathi, Average Message Overhead of Replica Control Protocols, *IEEE Symposium on Parallel and Distributed Computing*, 474-481, 1993.