An Optimal Voting Scheme for Minimizing the Overall Communication Cost in Replicated Data Management

Xuemin Lin * and Maria E. Orlowska †

December 14, 1995

Abstract

Quorum consensus methods have been widely applied to managing replicated data. In this paper, we study the problem of voting assignments for minimizing the overall communication cost of processing typical demands of transactions. This problem was left open, even restricted to a uniform network. In this paper, we shall show that for uniform networks, it can be solved by an efficient polynomial time algorithm.

Key words: concurrency control, replicated data management, optimization, quorum consensus method.

1 Introduction

The problem of managing replicated copies of data in a distributed database has received a great deal of attention [4,5,6,9,11,15] throughout the last decade. The main issue is to provide high data availability through data replication. Meanwhile, the replicated copies of data must be kept mutually consistent by synchronizing transactions at different sites so that a global serialization order can be ensured. To pursue mutual consistency, a *quorum consensus* (QC) method [2,4,5,12] has been proposed for managing replicated data.

In a QC method, an operation of a transaction issued at a site in a distributed database system can proceed only if permission is granted by a group of other sites storing the replicas of the data.

A basic QC method [2,4,11] can be described as follows:

*Department of Computer Science, University of Western Australia, Nedlands, WA 6907, Australia, E-mail: lxue@cs.uwa.oz.au

†Department of Computer Science, The University of Queensland, QLD 4072, Australia, Email: maria@cs.uq.oz.au

- A vote v_i (integer) is assigned to each site s_i.
- Two threshold values (integers) are assigned: one is referred to as read threshold Q_r (called *read quorum size*), and the other is referred to as write threshold Q_w (called *write quorum size*).
- *Two quorum intersection invariants* are assigned: $Q_r + Q_w > \sum_{i=1}^{n} v_i$, and $Q_w > \frac{\sum_{i=1}^{n} v_i}{2}$, where n is the number of sites.
- At each site s_i, the regulations for respectively forming a read quorum group S_i^r and a write quorum group S_i^w are as follows:

  add sites one by one to S_i^r (S_i^w) until the sum of votes in S_i^r (S_i^w) not less than Q_r (Q_w).
- Each read (write) operation should obtain permission from each site in S_i^r (S_i^w).

If a 2-phase locking mechanism is applied, a basic QC method will force, through the intersection invariants, the situation that a write and a read cannot take place simultaneously on different copies of the same data, and neither can two writes. Thus, mutual consistency can be maintained.

To resolve the limitations of a basic QC method, several other QC methods [1,10,3] have been proposed. Those approaches, including a basic QC approach, are associated with an assignment of a vote to each site. Moreover, to make each site bear equal responsibility for a read and a write, a number of *distributed* QC approaches [14,15] have been proposed. Those distributed QC approaches are based on a technique of *coteries* [5,7,8]. A recent research trend in developing new distributed QC approaches is to couple high data availability [15,16,13] with a low "communication" cost (to be defined in Section 2). Consequently, in a very reliable network, we should put our emphasis on reducing communication costs.

In this paper, we discuss only a basic QC method. Further, we restrict our interests in a static environment, that is, votes and quorum sizes are fixed *a priori.* The interested readers may refer to [6,9] for detailed discussions about *dynamic* QC methods.

In the rest of the paper, a basic QC method will be referred to as a BSQC method. A BSQC method is also called *majority voting* method in the literature if all v_i are the same. Otherwise, it is named a *weighted voting* method. A weighted voting method can potentially provide some benefits to matching the user requirements at each site, and then to reducing communication costs.

In a recent paper [11], Kumar and Segev show a tradeoff between overall communication costs and data availabilities. Several optimization problems have been proposed in

[11], as well as various optimization algorithms. Further, the problem of finding a BSQC method to minimize the overall communication cost for processing typical demands of transactions has been taken into account. However, without forcing the output to meet those data availability criteria in [11], this optimization problem was left open in [11], even restricted to a case where networks under consideration are "uniform" networks (see Section 2 for the definition). Only heuristics are claimed in [11]. We denote this problem by MCCU, which stands for "Minimizing Communication Cost over Uniform networks".

We shall show, in this paper, that MCCU can be solved in time $O(n^2 \log n)$ with respect to an improved transaction processing management model in comparison to that in [11]. Here, n is the number of sites in a network. Further, we show that the restriction of MCCU to the transaction processing management mode in [11] can be solved in time $O(n)$.

The rest of the paper is organized as follows. In Section 2, we present a formalization of the problem MCCU, together with the transaction processing management models. Section 3 gives solutions to MCCU. In Section 4, we present a discussion on a general network, and a brief analysis of the data availabilities provided by our solution. This is followed by conclusions.

2 A Formalization of MCCU

In this paper, we follow the model where replicated data is represented by multiple copies. We assume that the networks under consideration consist of n distributed processes (sites) which are fully connected. Each pair of processes can communicate only by passing messages, and do not share memory. We restrict our research, in this paper, to *uniform* networks where the *communication cost* between each pair of sites is the same c. By communication cost, we mean either the dollar cost of a unit data shipping or the time of a unit data shipping. We assume that each site knows the votes of the other sites. We also assume that the transactions are either a simple read operation or a simple write operation. (In Section 4, we will show how our result in the paper can be extended to cover a general case where a transaction may consist of several read and write operations.) Without loss of generality, we assume full replication in our environment; that is, a copy of each replicated object (data item) exists at all sites.

An assignment of votes $V = (v_1, v_2, ..., v_n)$ where each v_i is the vote of the site s_i, and quorum sizes Q_r and Q_w is *valid* if:

(2.1) $\sum_{i=1}^{n} v_i \geq Q_w$, $\sum_{i=1}^{n} v_i \geq Q_r$, and $Q_r + Q_w > \sum_{i=1}^{n} v_i$, $Q_w > \frac{\sum_{i=1}^{n} v_i}{2}$.

A valid assignment means that mutual consistency among multiple copies can be always guaranteed through BSQC. In the rest of the paper, we restrict our interests only to a valid assignment, that is, an assignment of V, Q_r and Q_w, whenever mentioned, always means a valid assignment.

A site s_j is a *key* site with respect to $(v_1, v_2, ..., v_n)$, Q_w, and Q_r, if $\sum_{i=1,i\neq j}^{n} v_i < Q_w$. Thus, by using BSQC, every write must get a vote from each key site.

We use a similar management model, as in [11], to perform a BSQC method for processing a transaction in a distributed environment. We assume that there is a *transaction manager* (TM) at each site. A write w (read r) is processed as follows.

The transaction manager (TM) at the issuing site s_j of w (r) acts as the coordinator. The coordinator site first obtains locks on the desired object in its local file. Then the coordinator assembles an appropriate write (read) quorum group using BSQC, and sends messages to the remote TMs in the write (read) quorum group, requesting them to either send their versions of the corresponding object (if the coordinator is not a key site), or to send only their replies to confirm that they have locked the corresponding records (if the coordinator is a key site). Each remote TM upon receiving a message must lock its own copy of the relevant object, and either

1. read them and send them to the coordinator if the coordinator is not a key site, or
2. send a confirmation about the implementation of lock to the coordinator if the coordinator is a key site.

After receiving the reply messages from all sites in the write (read) quorum group S_j^w (S_j^r), the coordinator will update the relevant object if necessary, and will run the transaction. Upon completion of the transaction, the coordinator will commit the transaction locally, release locks on the local copies, and send messages to the TMs at all other sites in the write (read) quorum group so that they can commit the transaction and release locks on their respective copies. For write operations, the new image of the object is also sent along with the commit message.

The traffic volume for a write w (or a read r) is $X_{1w} + X_{2w} + X_{3w}$ (or $X_{1r} + X_{2r} + X_{3r}$) if the coordinator is a key site, otherwise it is $X_{1w} + X_{2'w} + X_{3w}$ (or $X_{1r} + X_{2'r} + X_{3r}$). Here

- $X_{1r}(X_{1w})$ is the size of the request message from the coordinator to a remote site;
- X_{2r} (X_{2w}) is the size of the reply message from the remote site to the coordinator if the coordinator site is the key site;
- $X_{2'r}$ $(X_{2'w})$ is the size of the reply message from the remote site to the coordinator site if the coordinator site is not the key site;

- X_{3r} is the size of the release lock and commit message from the coordinator to the remote site (for read operation); and

- X_{3w} is the size of the update record, release lock, and commit message from the coordinator to the remote site.

Note that for the same transaction r (w), $X_{2'r}$ is usually larger than X_{2r}, and $X_{2'w}$ is usually larger than X_{2w}.

For the same transaction, the size of each reply message from a remote site may be different with respect to different sites if the coordinator is not a key site. As noted in [11], it is usually difficult to predict the difference among those reply messages. Here, we use the same approximate treatment as in [11] by viewing them as the same $X_{2'r}$ ($X_{2'w}$). In [11], the authors assume that a transaction from a key site is processed in the same way in which a transaction from a non-key site is processed, that is, $X_{2r} = X_{2'r}$ and $X_{2w} = X_{2'w}$. We drop this restriction in the paper, since a key site keeps all the update information, and we don't need any remote site to send its current version of a relevant object of a file for processing a transaction from a key site.

We can assume that the statistics information obtained by us is as follows. With respect to each object (data item), we record, at each site, how many writes w are issued, the frequency f_w of each write w, and the values of X_{1w}, X_{2w}, $X_{2'w}$, and X_{3w} for each w. Also, we record how many reads r are be issued at each site, the frequencies f_r of each r, and the values of X_{1r}, X_{2r}, $X_{2'r}$, and X_{3r} for each r.

Thus, with respect to a data item, at each site s_j, let:

- r_j denote the summation of all $f_r(X_{1r}+X_{2r}+X_{3r})$ for all reads r from s_j, representing the total data volume of read traffic from s_j in the case that s_j acts as a key site, and

- r'_j denote the summation of all $f_r(X_{1r} + X_{2'r} + X_{3r})$ for all reads r from s_j, representing the total data volume of read traffic from s_j in the case that s_j does not act as a key site, and

- w_j denote the total data volume of write traffic from s_j in the case that s_j is assigned as a key site, and

- w'_j denote the total data volume of write traffic from s_j in the case that s_j is not assigned as a key site.

Note that $r'_i \geq r_i$ and $w'_i \geq w_i$, since each $X_{2'r} \geq X_{2r}$ and each $X_{2'w} \geq X_{2w}$.

To minimize the overall communication cost, we need only to consider the following restricted BSQC method:

At each site s_i, every S_i^r (S_i^w) should be formed by firstly choosing s_i itself.

The inclusion of a local vote can always lead to an access of a fewer number of remote sites. Therefore, in this paper, we study only the restricted BSQC method.

Now, the problem is that for each given data item, we would like to find an optimal voting scheme. Suppose that a data item is given, and $L = \{r_i, r'_i, w_i, w'_i : 1 \le i \le n\}$ with respect to the data item is given such that for each i, $r'_i \ge r_i$ and $w'_i \ge w_i$, an assignment of votes $V = (v_1, ..., v_n)$, Q_r, and Q_w is given. In the application of BSQC, at each site s_i, there is an optimal read quorum group S^r_{i,V,L,Q_r,Q_w} such that the total communication cost for processing the all reads issued at s_i is minimized. Clearly, there also exist a optimal write quorum group S^w_{i,V,L,Q_r,Q_w} with the minimum total communication cost for processing the all writes issued at s_i. The problem of MCCU can be expressed precisely as follows:

INSTANCE: given $L = \{r_i, r'_i, w_i, w'_i : 1 \le i \le n\}$ such that for each i, $r'_i \ge r_i$ and $w'_i \ge w_i$.

QUESTION: find an assignment of $V = (v_1, ..., v_n)$, Q_w and Q_r, such that the following value is minimized:

$$\sum_{i=1}^{n}(|S^r_{i,V,L,Q_r,Q_w}| - 1)c(\delta(v_i, Q_w)r_i + (1 - \delta(v_i, Q_w))r'_i) +$$
$$\sum_{i=1}^{n}(|S^w_{i,V,L,Q_r,Q_w}| - 1)c(\delta(v_i, Q_w)w_i + (1 - \delta(v_i, Q_w))w'_i). \qquad (1)$$

Here for each i, $\delta(v_i, Q_w) = 1$ if s_i is a key site with respect to V and Q_w, otherwise $\delta(v_i, Q_w) = 0$. (1) is the overall communication cost for processing the transactions on a given data item by using BSQC with the quorum groups S^r_{i,V,L,Q_r,Q_w} and S^w_{i,V,L,Q_r,Q_w} at each site. (1) is referred to as the *cost* of the assignment of V, Q_w and Q_r with respect to L. As mentioned earlier, c is the communication cost of a unit data shipping along a link.

Note that in this paper, we study a more general optimization problem than the optimization problem in [11]. In [11], they assume that for each i, $r_i = r'_i$ and $w_i = w'_i$, and for each s_i, each formed S_i^r and S_i^w have the properties that $\sum_{j \in S_i^r} v_j = Q_r$ and $\sum_{j \in S_i^w} v_j = Q_w$. We may expect that the overall communication cost with respect to a solution of MCCU is never greater than that with respect to a solution of the restricted MCCU in [11], since the problem domain of MCCU is larger than that of the restricted MCCU.

3 An Efficient Solution to MCCU

Obviously, a trivial exhaustive search for solving MCCU will be exponentially time bounded. In this section, we present an $O(n^2 \log n)$ algorithm OPT for solving the problem MCCU.

An assignment A of votes $V = (v_1, ..., v_n)$, and quorum sizes Q_r and Q_w is a *key site based assignment* if there is a positive integer l, such that

- $v_{j_i} = n - l + 1$ for $1 \leq i \leq l$, and
- $v_{j_i} = 1$ for $l + 1 \geq i \geq n$, and
- $Q_r = n - l + 1$ and $Q_w = l(n - l + 1)$.

Here, $(v_{j_1}, v_{j_2}, ..., v_{j_n})$ is a permutation of $(v_1, v_2, ..., v_n)$. $KEY = \{s_{j_x} : 1 \leq x \leq l\}$ is called the *key site set* of A, since it can be verified that each site in KEY is a key site. Moreover, for any $L = \{r_i, r'_i, w_i, w'_i : 1 \leq i \leq n\}$, it can be verified that in the application of BSQC on the top of A, the optimal read quorum groups and the optimal write quorum groups have the following properties:

- for $1 \leq i \leq l$, $S^r_{j_i}$ is always $\{s_{j_i}\}$ and $S^w_{j_i,V,L,Q_r,Q_w} = KEY$, and
- for $l + 1 \geq i \geq n$, $S^r_{j_i,V,L,Q_r,Q_w}$ consists of s_{j_i} and a site in KEY, and $S^w_{j_i,V,L,Q_r,Q_w} = \{s_{j_i}\} \cup KEY$.

Thus, the (communication) cost, as described in (1), of A with respect to L can be re-written as:

$$\begin{aligned}
& c(\sum_{s_i \in KEY} (|KEY| - 1)w_i + \sum_{s_i \notin KEY} (r'_i + |KEY|w'_i)) \\
= \ & c(\sum_{i=1}^{n}(r'_i + |KEY|w'_i) - \sum_{s_i \in KEY} (r'_i + w'_i + (|KEY| - 1)(w'_i - w_i))). \qquad (2)
\end{aligned}$$

Note that a key site based assignment is determined by its key site set; and an assignment of votes and quorum sizes with some key sites is not necessarily a key site based assignment. Given a key site based assignment A, in order to force the BSQC approach to always firstly assemble an optimal read quorum group and an optimal write quorum group at each site, we can implement BSQC as follows:

> After choosing the local site, we gradually add a site with the largest vote within the remaining sites to a (read or write) quorum group until the sum of the votes not less than the (read or write) quorum size.

The algorithm OPT will choose an appropriate key site based assignment as a solution to MCCU. The algorithm OPT consists of the following two steps:

Step 1: For $1 \leq k \leq n$, let KEY_k consist of k sites s_i whose $r'_i+w'_i+(k-1)(w'_i-w_i)$ are the first k's largest values among sites s_i $(1 \leq i \leq n)$, that is, $r'_i+w'_i+(k-1)(w'_i-w_i) \geq r'_j+w'_j+(k-1)(w'_j-w_j)$ if $s_i \in KEY_k$ and $s_j \notin KEY_k$. Then based on each KEY_k, a key site based assignment A_k of votes and quorum sizes can be determined such that KEY_k is the key site set of A_k. Go to Step 2.

Step 2: For $1 \leq k \leq n$, find a A_k such that its cost is minimized within $\{A_i : 1 \leq i \leq n\}$. Output A_k.

In the algorithm OPT, the most expensive procedure is to find the first k's largest values of $r'_i + w'_i + (k-1)(w'_i - w_i)$, for each k, among all sites s_i. Here, we apply a simple implementation of this procedure. To find the first k's largest values for each k, we first carry out sorting. Thus, Step 1 takes $O(n^2 \log n)$. Meanwhile, Step 2 takes only $O(n)$. It follows that Algorithm OPT runs in $O(n^2 \log n)$.

Clearly, we can manage to use only $O(n)$ space to implement the algorithm OPT, by storing only the optimal key site based assignment with the key site set size range from 1 to k for $k \leq n$.

For example, in a uniform network consisting of 4 sites and $c = 1$, let:

> $r'_1 = 100, r_1 = 50, w'_1 = 10, w_1 = 8$, $r'_2 = 60, r_2 = 40, w'_2 = 5, w_2 = 4$, $r'_3 = 10, r_3 = 5, w'_3 = 5, w_3 = 4$, $r'_4 = 70, r_4 = 30, w'_4 = 10, w_4 = 8$.
> After Step 1 in the algorithm OPT, A_1 is the key site based assignment with key site set $\{s_1\}$; the key site set KEY_2 of A_2 is $\{s_1, s_4\}$; the key site set KEY_3 of A_3 is $\{s_1, s_2, s_4\}$; the key site set KEY_4 of A_4 is $\{s_1, s_2, s_3, s_4\}$.
> In Step 2, we use (2) to compute the costs of A_1, A_2, A_3, and A_4. They are respectively 160, 106, 65, 72. So, we choose A_3 as the output of the algorithm OPT.

We now prove that the algorithm OPT gives a solution to the problem MCCU. Our proof consists of the following aspects:

1. The replacement of an assignment of votes and quorum sizes, which has a set π of key sites, by the key site based assignment with π as its key site set will always lead to a smaller total communication cost for processing a given set of transactions.

2. The replacement of an assignment of votes and quorum sizes, which does not have a key site, by any key site based assignment with a single key site will always lead to a smaller total communication cost.

3. The output of our algorithm is the optimal key site based assignment.

First, we show the following two important facts.

Lemma 1 *Suppose that votes $V = (v_1, ..., v_n)$, and quorum sizes Q_r and Q_w are assigned such that s_i is not a key site. Then $v_i < Q_r$ and $v_i < Q_w$. (In other words, either a read or a write from a non-key site must access at least one remote site.)*

Proof: Since v_i is not a key site, we have that

$$Q_w \leq \sum_{j=1, j \neq i}^{n} v_j. \tag{3}$$

This together with $Q_w + Q_r > \sum_{j=1}^{n} v_j$ implies that $Q_r > v_i$.

(3) together with $Q_w > \frac{\sum_{i=1}^{n} v_i}{2}$ also implies that $Q_w > v_i$. One can verify this by a simple calculation. □

Lemma 2 *Suppose that an assignment of $V = (v_1, ..., v_n)$, Q_r, and Q_w is given, such that π is the set of all key sites. Let S_i^w is a formed write quorum group by BSQC at each site s_i. Then $\pi \subseteq S_i^w$.*

Proof: From the definitions of a key site and a write quorum group, this Lemma immediately follows. □

Next we prove the first aspect.

Lemma 3 *Suppose that $A1$ is an assignment of $V = (v_1, ..., v_n)$, Q_r, and Q_w. π is the set of key sites with respect to V, Q_r, and Q_w. Further, suppose that A is the key site based assignment with π as its key site set. Then the cost of A is smaller than or equal to the cost of $A1$.*

Proof: From Lemmas 1 and 2, it follows that the cost of any assignment, with π as the set of key sites, of votes $V = (v_1, ..., v_n)$, and quorum sizes Q_r and Q_w, is larger than or equal to $c(\sum_{s_i \in \pi}(|\pi| - 1)w_i + \sum_{s_i \notin \pi}(r_i' + |\pi|w_i'))$. This proves the Lemma. □

From Lemma 1, we can prove the second aspect.

Lemma 4 *In any assignment A of votes and quorum sizes, if there are no keys, then the cost of any key site based assignment A_1, whose key site set consists only one site, is smaller than or equal to the cost of A.*

Proof: From Lemma 1, it follows that the cost of A is not smaller than $c(\sum_{i=1}^{n}(r'_i + w'_i))$. Meanwhile, the cost of a key site based assignment, whose key site set consists of only site s_j, is: $c(w_j + \sum_{i=1, i\neq j}^{n}(r'_i + w'_i))$. Note that each $w_j \leq w'_j$. The Lemma follows immediately. □

From Lemmas 3 and 4, it follows that we need only to choose an appropriate key site based assignment as a solution to MCCU. Next, we show the third aspect.

Lemma 5 *Suppose that $L = \{r_i, r'_i, w_i, w'_i : 1 \leq i \leq n\}$ is given. Among key site based assignments of votes and quorum sizes with k key sites, a key site based assignment A_k, such that the key site set consists of those k key sites s_i whose $r'_i + w'_i + (k-1)(w'_i - w_i)$ are the first k's largest values, has the minimal cost.*

Proof: To prove this Lemma, we need only prove the following fact.

> Let A^1 and A^2 are two key site based assignments respectively with k key sites. Suppose that KEY^1 and KEY^2 are the corresponding key site sets of A^1 and A^2 such that KEY^1 consists of π (a set of $k-1$ sites) and s_i, KEY^2 consists of π and s_j, $r'_i + w'_i + (k-1)(w'_i - w_i) \geq r'_j + w'_j + (k-1)(w'_j - w_j)$. Then, the cost of A^1 is not greater than that of A^2.

By using (2), we may immediately verify this fact. □

From Lemmas 3, 4, 5, and the algorithm OPT, it follows:

Theorem 1 *Algorithm OPT gives a solution to the problem MCCU.*

A Remark about MCCU:

If we apply the same transaction management model as that in [11], we may speed up our algorithm OPT for solving the problem MCCU. In that transaction management model it is assumed that a transaction from a key site is processed in the same way as those from a non-key site. That is, for at each s_i, $X_{2r} = X'_{2r}$ and $X_{2w} = X'_{2w}$. This implies that for each s_i, $r_i = r'_i$ and $w_i = w'_i$. We use SMCCU to denote the problem, MCCU, restricted to the transaction management model in [11]. Here, SMCCU stands for "Simple MCCU".

All the Lemmas and Corollaries, proven earlier, still hold for solving SMCCU. Further, we are able to characterize explicitly how many key sites we need and what kind of site can be a key site.

Lemma 6 *Suppose that A is an arbitrary key site based assignment of votes and quorum sizes, and KEY is its key site set. Then, a key site based assignment A_1, with one of the following two properties, will never lead to a larger communication cost to that of A:*

1. *the key site set* KEY_1 *of* A_1 *is* $KEY \cup \{s_{i_0}\}$ *where* $s_{i_0} \notin KEY$ *and* $r_{i_0} + w_{i_0} - \sum_{j=1}^{n} w_j \geq 0$*, or*

2. *the key site set* KEY_1 *of* A_1 *is* $KEY - \{s_{i_0}\}$ *where* KEY *has at least two elements,* $s_{i_0} \in KEY$*, and* $r_{i_0} + w_{i_0} - \sum_{j=1}^{n} w_j < 0$.

Proof: Noting the formula (2), we have that the communication cost with respect to A is:

$$c(\sum_{i=1}^{n}(r_i + |KEY|w_i) - \sum_{s_i \in KEY}(r_i + w_i)), \tag{4}$$

and the communication cost with respect to A_1 is:

$$c(\sum_{i=1}^{n}(r_i + |KEY_1|w_i) - \sum_{s_i \in KEY_1}(r_i + w_i)), \tag{5}$$

In case that A_1 has the property 1, the formula (5) can be re-written as:

$$c(\sum_{i=1}^{n}(r_i + |KEY|w_i) - \sum_{s_i \in KEY}(r_i + w_i) - (r_{i_0} + w_{i_0} - \sum_{j=1}^{n} w_j)). \tag{6}$$

It follows that the Lemma holds for a key site based assignment A_1 with the property 1.

In case that A_1 has the property 2, the formula (5) can be re-written as:

$$c(\sum_{i=1}^{n}(r_i + |KEY|w_i) - \sum_{s_i \in KEY}(r_i + w_i) + (r_{i_0} + w_{i_0} - \sum_{j=1}^{n} w_j)). \tag{7}$$

It follows that the Lemma holds for a key site based assignment A_1 with the property 2. □

Thus, we can obtain a more efficient algorithm OPTS, than the algorithm OPT, to solve SMCCU. The algorithm OPTS proceeds as follows, to find an appropriate key site based assignment:

(A) If there are some sites such that $r_i + w_i - \sum_{j=1}^{n} w_j \geq 0$, the algorithm will choose those sites s_i, with $r_i + w_i - \sum_{j=1}^{n} w_j \geq 0$, to form a site set KEY, and then output the key site based assignment with KEY as its key site set. Otherwise go to (B).

(B) The algorithm will choose the site s_i such that $r_i + w_i - \sum_{j=1}^{n} w_j$ is maximized. And then it will output the key site based assignment with $\{s_i\}$ as its key site set.

It is clear that we can scan all sites only once to implement the algorithm OPTS. This means that the algorithm OPTS takes $O(n)$.

4 Further Discussions on Optimal Voting Scheme

The problem of finding a BSQC method to minimize the overall communication cost for transaction processing in a general network appears difficult. The same technique, developed in this paper, cannot be applied to the optimization problem with respect to a general network. We show, as follows, that a key site based assignment is not always the best choice in a general network.

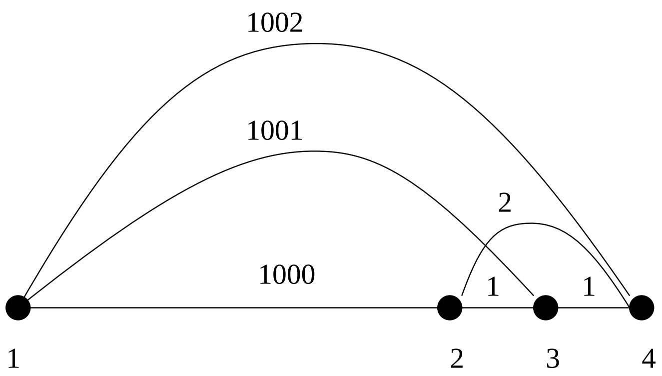

Figure 1: a general network

Suppose that a network is given as illustrated in Figure 1, where the number in a link indicates the communication cost of a unit data shipping along this link. We also assume that:

$$w'_1 = w_1 = 1 \text{ and } r'_1 = r_1 = 1000; \text{ for } 2 \leq i \leq 4,\ w'_i = w_i = 50 \text{ and } r'_i = r_i = 2.$$

It can be immediately verified that any key site based assignment of votes and quorum sizes is worse than following assignment.

- $v_1 = 4$ and $v_2 = v_3 = v_4 = 1$,
- $Q_r = 3$, $Q_w = 5$,
- $S_1^r = \{1\}$, $S_1^w = \{1,2\}$, $S_2^r = \{2,3,4\}$, $S_2^w = \{2,1\}$, $S_3^r = \{2,3,4\}$, $S_3^w = \{3,1\}$, $S_4^r = \{2,3,4\}$, $S_4^w = \{4,1\}$.

So we should develop new techniques to investigate the optimization problem in a general network.

In the preceding discussion of the MCCU problem, we made the assumption that each transaction is either a single read or a single write. In most application environments, a transaction may consist of several reads and several writes, and thus, an operation is not always associated with a commit operation. However, after the completion of each operation at the coordinator site, messages are always sent from the coordinator to the remote sites in a quorum group to ask them to either downgrade (upgrade) its lock or release its lock for commitment. Approximately, we can view them as a same size message, and then record it as the commitment message in our formalization.

> For example, in a network with 3 sites. A transaction is issued from site 3 which consists of two operations (a write operation is followed by a read) on the same data item. The write quorum group consists of all 3 sites, and the read quorum group consists of sites 3 and 2. After site 3 completes the write, it sends message to site 2 together with new image to ask it to downgrade the write lock for processing a read. Then after the completion of the read (also the transaction), site 3 will send a commitment message to site 2 and site 1 to do the commitment (note the message to site 1 should also contain the new image.)
>
> Thus, associated with the write operation w there are two different messages after the completion, one is sent to site 2, and another is sent to site 1. We approximately view them as the same size message, and record it as X_{3w} in our preceding formalization.

The major disadvantages with the solution produced by the algorithm OPT are:

- The communication traffic to the key sites and local processing at the key sites will be very high in comparison with those at non-key sites.
- A key site failure will stop the processing of any write in the whole network, though it can tolerate non-key site failures for a write and a read.
- The failures of all key sites will also stop the processing of any read in the whole network, though it can tolerate some key site failures.

The above disadvantages are the price that we have to pay for minimizing the total communication cost. However, we may overcome the first disadvantage by providing powerful computers at the key sites and high-bandwidth lines connecting the key sites to ensure fast computation. We can also maintain a high availability of key sites to reduce the site failures.

Assume that in an application environment the total write load is much lower than the read load at each site, and in the solutions produced by OPT there are $f(n)$ key sites where $f(n) \rightarrow \infty$ when $n \rightarrow \infty$. Then those solutions also have an asymptotically *high site resilience* [13,15] with respect to a read.

5 Conclusion

In this paper, we investigate the quorum consensus methods for managing replicated data in distributed database systems. The network environment considered in this paper is a uniform network with n sites. We present an algorithm, $O(n^2 \log n)$, to produce an optimal solution to the problem of finding a BSQC method to minimize the overall communication cost for transaction processing. This takes the form of an improved transaction management model in comparison with that in [11]. Meanwhile, we also show that the optimization problem, restricted to the transaction management model in [11], can be solved in $O(n)$. A possible future study may be carried out through a general network.

Acknowledgement

The work of the first named author was partially supported by IRG at UWA, while the work of the second named author was partially supported by DSTC. The authors greatly thank the anonymous referees for many good comments.

References

[1] D. Agrawal and A. El Abbadi, An Efficient and Fault-Tolerant Algorithm for Distributed Mutual Exclusion, *Proceedings of the Eight Annual ACM Symposium on Principles of Distributed Computing*, 193-200, 1989.

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

[3] S. Y. Cheung, M. Ammar and M. Ahamad, The Grid Protocol: A High Performance Scheme for Maintaining Replicated Data, *Proceedings of the Sixth International Conference on Data Engineering*, 438-455, 1990.

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

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

[6] M. Herlihy, Dynamic Quorum Adjustment for Partitioned Data, *ACM Transactions on Database Systems*, 12(2), 170-194, 1987.

[7] T. Ibaraki and T. Kameda, Boolean Theory of Coteries, *3rd IEEE Symposium on Parallel and Distributed Processing*, 150-157, 1991.

[8] T. Ibaraki and T. Kameda, A Theory of Coteries: Mutual Exclusion in Distributed Systems, *IEEE Transactions on Parallel and Distributed Systems*, 4(7), 779-794, 1993.

[9] S. Jajodia and D. Mutchler, Dynamic Voting Algorithms for Maintaining the Consistency of a Replicated Database, *ACM Transactions on Database Systems*, 15(2), 230-280, 1990.

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

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

[12] L. Lamport, The Implementation of Reliable Distributed Multiprocess Systems, *Computer Networks*, 2, 95-114, 1978.

[13] X. Lin and M. Orlowska, A Highly Fault-Tolerant Quorum Consensus Method for Managing Replicated Data, *COCOON'95*, Lecture Notes in Computer Science, 959, Springer-Verlag, 171-180, 1995.

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

[15] 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.

[16] M. Spasojevic and P. Berman, Voting as the Optimal Static Pessimistic Scheme for Managing Replicated Data, *IEEE Transaction on Parallel and Distributed Systems*, 5(1), 64-73, 1994.