Multi-Dimensional Voting: A General Method for Implementing Synchronization in Distributed Systems*

Shun Yan Cheung Mustaque Ahamad Mostafa H. Ammar
School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

Abstract

We introduce a new concept, multi-dimensional voting, in which the vote and quorum assignments are k-dimensional vectors of non-negative integers and each dimension is independent of the others. Multi-dimensional voting is more powerful than traditional weighted voting because it is equivalent to the general method for achieving synchronization in distributed systems which is based on coteries (set of groups of nodes) but its implementation is easier than coteries. We describe an efficient algorithm for finding a multi-dimensional vote assignment for any given coterie and show examples of its use. We also show how multi-dimensional voting can be used to easily implement novel algorithms for synchronizing access to replicated data or to ensure mutual exclusion. These algorithms cannot be implemented by traditional weighted voting.

1 Introduction

Distributed systems offer many advantages, including resource sharing and fault-tolerance. The latter can be achieved by replicating a resource at nodes with independent failure modes. Replication can also improve performance when load is shared among the nodes that have instances of a resource. In many applications, users need to synchronize access to shared resources. For example, when data is replicated to improve its availability, updating the file requires mutually exclusive access. This is necessary for maintaining the consistency of the data. The synchronization technique should work in the presence of node and communication failures.

An operation that requires mutual exclusion can be executed if permission can be obtained from a group of nodes. In general, a node can execute the operation if permission can be obtained from any one group in a set of intersecting groups [1]. Such a set is called a coterie in [2]. For reading and writing of replicated data when several readers are allowed to access the data concurrently, read and write coteries can be defined in a similar way [3]. Another well-known synchronization method is weighted voting [4] which is a generalization of the majority consensus method [5]. In voting, each node is assigned a number of votes and each operation must obtain a pre-defined quorum of votes before it is allowed to execute to completion. Voting can be used for achieving mutual exclusion and synchronizing reading and writing of replicated data. In mutual exclusion, each operation must obtain a majority of the votes assigned before it can proceed. In reading and writing, the read and write quorums must be such that their sum is more than the total number of votes and the write quorum is at least a majority of all votes.

Voting is appealing because it is flexible and can be easily implemented. Each node in voting stores its assigned vote and when it wants to execute an operation that requires q votes, it communicates with other nodes to request their votes. The execution of the operation can proceed if the sum of the votes received is at least q. In contrast, in a system that uses coteries, operations must know all the groups of the coterie and test if the nodes that responded positively to its request form a group of the coterie. Voting is also more flexible. Adding or removing a node requires only a change of the quorum and assigning the proper number of votes to the new node. In a coterie-based system, adding and removing a node may cause the addition and deletion of numerous groups. However, García-Molina and Barbara proved in [2] that the method of coteries is more general than voting by showing coteries which cannot be obtained from any vote assignment. Coteries that are not obtained from vote assignments can be used to achieve better performance by reducing the number of messages. For example, structured coteries as those used in the methods described in [6, 7] have lower communication cost and they cannot be implemented by voting.

We present in this work a new voting based method that is as powerful as the method of coteries and has the flexibility and ease of implementation of voting. In multi-dimensional voting, the vote assignment to each node and the quorums are k-dimensional vectors of non-negative integers. Each dimension of the vote and quorum assignment is similar to voting and the quorum requirements in different dimensions can be combined in a number of ways.
This makes multi-dimensional voting more powerful than standard voting. We will discuss a number of applications which can be implemented with multi-dimensional voting but not with standard voting.

We show that every coterie can be represented by a multi-dimensional vote assignment and present an efficient algorithm for finding one. The use of the algorithm is demonstrated in finding multi-dimensional vote assignments for coteries that cannot be obtained from standard vote assignments. We also discuss how reading and writing of replicated data can be synchronized using multi-dimensional voting.

The paper is organized as follows. In Section 2, we will introduce the concept of a multi-dimensional vote assignment and in Section 3, we present an algorithm for finding multi-dimensional vote and quorum assignments for sets of groups with a certain property which is satisfied by coteries. Sections 4 and 5 discuss the use of multi-dimensional voting for mutual exclusion and reading and writing of replicated data respectively. We conclude the paper in Section 6.

1.1 Related Work

Coteries and voting have been used to synchronize access to replicated data and the methods are called replica control protocols. Related work on these protocols include many dynamic replica control methods that are derived from voting [8, 9, 10, 11, 12] and these methods achieve very high data availability. Available copies and regeneration methods, such as the ones described in [13] and [14], achieve even higher data availability using the same number of copies but data consistency cannot be guaranteed if the network can partition. To reduce storage space for copies, the methods described in [15] and [16] can be used, but data availability will also be compromised. The mutual exclusion method presented in [6] uses coteries that are derived from a binary tree structure. A comprehensive survey of replica control protocols is presented in [17].

The problem of enumerating coteries so that performance can be optimized by choosing the best one has been addressed by several researchers. In [2], an algorithm is described that can be used to generate a subset of the mutual exclusion coteries which includes all coteries obtained from vote assignments. The authors presented an algorithm in [3] to generate all vote and quorum assignments that need to be considered in optimizing reading and writing of replicated data. The method presented in [18] generates a subset of the mutual exclusion coteries obtained from vote assignments.

Optimization using availability as the performance measure has been considered in a number of works. Barbara and Garcia-Molina showed in [19] that the vote assignment which allocates one vote to each node will maximize availability for mutual exclusion if the nodes are uniform. In a related work [20], Ahamad and Ammar studied availability and response time of read and write operations for systems of uniform nodes and in [21], the authors presented a scheme that can improve response time through a higher degree of load sharing. The performance of the available copies replica control protocol and its variants is studied in [22] and [23].

2 Multi-Dimensional Voting

We consider a distributed system of \( N \) nodes which are numbered as 1, 2, \ldots, \( N \). In multi-dimensional (MD) voting, the vote value assigned to a node and the quorum are \( k \)-dimensional vectors of non-negative integers. Formally, the MD vote assignment \( V_{N,k} \) is a \( N \times k \) matrix where \( v_{i,j} \) represents the vote assignment to node \( i \) in the \( j^{th} \) dimension and \( v_{i,j} \geq 0 \) for \( i = 1,2,\ldots,N \) and \( j = 1,2,\ldots,k \). The votes assigned in the various dimensions are independent of each others. The quorum assignment \( q = (q_1,q_2,\ldots,q_k) \) is a \( k \)-dimensional integer vector, where \( q_j > 0 \), for \( j = 1,2,\ldots,k \). In addition, a number \( \ell \), \( 1 \leq \ell \leq k \), is defined which is the number of dimensions of vote assignments for which the quorum must be satisfied. Thus, there are two levels of requirements: vote and dimension level. At the vote level, the number of votes must be greater than or equal to the quorum requirement in the same dimension and at the dimension level, the number of dimensions for which a quorum is collected must be greater than or equal to \( \ell \). As we show in the next section, this extra level of flexibility makes MD-voting more powerful than standard voting. We denote MD-voting with quorum requirement in \( \ell \) of \( k \) dimensions as MD(\( \ell \),\( k \))-voting and the term SD-voting (single dimensional voting) will refer to the standard voting method described in [4]. In fact, MD(1,1)-voting is the same as SD-voting.

Synchronization methods developed from MD-voting operate in a similar manner as SD-voting. Each node stores its vote which consists of \( k \) integers and each operation has a quorum requirement for each dimension and the value of \( \ell \). An operation requests permission from the nodes by sending a voting request to them. When a node receives a vote request, it votes reject if it wants to disallow the operation to proceed (e.g., due to locking conflict) or replies with its vote in all dimensions. Each operation maintains \( k \) independent variables which accumulate the votes received in each dimension. When a response containing a vote is received, the operation adds the vote in each dimension to the appropriate variable and when the sums in at least \( \ell \) variables are greater than or equal to the quorum in the corresponding dimensions, the operation can proceed.

Each node must store \( k \) integers and the voting messages used will also contain all the integers. When a large number of dimensions is used, the voting messages can be long and in the next section, we will present a method for finding MD vote and quorum assignments that use a relatively small number of dimensions.
3 Finding a Multi-Dimensional Vote Assignment

3.1 Definitions and Notation

Let \( U = \{1, 2, \ldots, N\} \) be the universe set of all nodes and we will refer to sets of nodes as groups. A set of groups \( Q \) has the minimality property [2] if,

\[
\forall G, H \in Q : \ G \subseteq H
\]

The synchronization requirements define what groups are included in the set. For example, if mutual exclusion is desired, this set is called a coterie and any two of its members must have a non-empty intersection (see Section 4).

A number of the sets that have the minimality property can be represented by SD-voting. Each node \( i \) in SD-voting is assigned \( v_i \) votes \((1 \leq i \leq N)\) where \( v_i \) is a non-negative integer and a quorum \( q \) is defined, such that nodes in each group of the set have at least \( q \) votes. Specifically, with the vote assignment \( \gamma = (v_1, v_2, \ldots, v_N) \), the members of the set defined by \( (\gamma, q) \) are tight groups of nodes which have at least \( q \) votes. A group \( G \) is tight with respect to quorum \( q \) if,

\[
\sum_{g \in G} v_g \geq q \quad \text{and,}
\]

any proper subset of \( G \) has less than \( q \) votes

The set of tight groups \( Q \) defined by \( (\gamma, q) \) is,

\[
Q = \{ G \mid G \text{ is a tight group with respect to quorum } q \}
\]

and this set has the minimality property since if there would exist \( G, H \in Q \) such that \( G \subseteq H \), then \( H \) would not be tight. The vote and quorum assignment \( (\gamma, q) \) defines a unique set of tight groups which has the minimality property. For instance, the vote assignment \( (1,1,1) \) to a three node system and the quorum requirement \( 2 \) votes defines the set of tight groups \( \{(1,2), (1,3), (2,3)\} \). The same set can be represented by the vote assignment \( (2,2,3) \) and \( q = 4 \) and hence a set may not have a unique vote and quorum assignment.

An MD(\( \ell, k \)) vote and quorum assignment also defines a unique set of tight groups in a similar manner as SD-voting. A group \( G \) is a tight group in MD(\( \ell, k \))-voting with respect to quorum requirement \( q \) if

\[
\sum_{g \in G} v_{g,j} \geq q_j \quad \text{for } \ell \text{ distinct dimensions } j_1, j_2, \ldots, j_\ell \text{ and,}
\]

any proper subset of \( G \) satisfies quorum requirement in strictly less than \( \ell \) dimensions

The set \( Q_{\ell,k}(V_{N,k}, q_{\ell,k}) \) of tight groups represented by the MD(\( \ell, k \)) vote and quorum assignment \( (V_{N,k}, q_{\ell,k}) \) is,

\[
Q_{\ell,k}(V_{N,k}, q_{\ell,k}) = \{ G \mid G \text{ is a tight group in MD(\( \ell, k \))-voting with respect to } q \}
\]

Similar to SD-voting, the same set of tight groups can be represented by different MD(\( \ell, k \)) vote and quorum assignments and the groups are also minimal. The set of tight groups for the special cases where \( \ell = 1 \) (any dimension) and \( \ell = k \) (all dimensions) can be given as follows,

\[
Q_{1,k}(V_{N,k}, q_{1,k}) = \{ G \mid G \text{ is a tight group such that:} \exists j; 1 \leq j \leq k : \sum_{g \in G} v_{g,j} \geq q_j \}
\]

\[
Q_{k,k}(V_{N,k}, q_{k,k}) = \{ G \mid G \text{ is a tight group such that:} \forall j; 1 \leq j \leq k : \sum_{g \in G} v_{g,j} \geq q_j \}
\]

In MD(\( 1,k \))-voting, an operation can proceed if quorum is available in any dimension and in MD(\( k,k \))-voting, quorum requirements in all dimensions must be satisfied. For MD(\( 1,k \))-voting, we can also write,

\[
Q_{1,k}(V_{N,k}, q_{1,k}) = \{ G \mid G \in \bigcup_{j=1}^k C_j \text{ and } \forall H \in \bigcup_{j=1}^k C_j : H \not\subseteq G \}
\]

where \( C_j \) is the set of tight groups defined by the \( j^{th} \) dimension of vote and quorum assignment, i.e, \( Q_{1,k}(V_{N,k}, q_{1,k}) \) is all the minimal groups in \( \bigcup_{j=1}^k C_j \).

Table 1 presents a two-dimensional vote and quorum assignment to a system of four nodes. The sets \( C_1 \) and \( C_2 \) are the sets of tight groups corresponding to the first and second dimension of the MD vote and quorum assignment, respectively.

<table>
<thead>
<tr>
<th>( V_{1,2} )</th>
<th>( q_1 = (2,3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1</td>
<td>( C_2 = (2,3) )</td>
</tr>
<tr>
<td>1 1</td>
<td></td>
</tr>
<tr>
<td>1 1</td>
<td></td>
</tr>
<tr>
<td>0 2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( C_1 )</th>
<th>( (1,2), (1,3), (2,3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_2 )</td>
<td>( (1,4), (2,4), (3,4), (1,2,3) )</td>
</tr>
</tbody>
</table>

\[
Q_{1,2}(V_{4,2}, q_{1,2}) = \{(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)\}
\]

\[
Q_{2,2}(V_{4,2}, q_{2,2}) = \{(1,2,3), (1,2,4), (1,3,4), (2,3,4)\}
\]

Table 1: An example of multi-dimensional vote assignment

3.2 The Existence of MD Vote Assignments

We show that any set \( Q \) that has the minimality property can be represented by an MD(\( 1,k \)) vote and quorum assignment.
Lemma 3.1: Let \( Q \) be a set of groups such that
\[
\forall G, H \in Q : \quad G \nsubseteq H
\]
Then \( Q \) can be represented as an \( MD(1,k) \) vote and quorum assignment where \( k = |Q| \).

Proof: Let \( Q = \{ G_1, G_2, \ldots, G_k \} \) so that \( |Q| = k \). We construct the following \( k \)-dimensional vote assignment: the vote value of node \( i \) in the \( j \)-th dimension, \( i = 1, 2, \ldots, N \) and \( j = 1, 2, \ldots, k \) is given by,
\[
v_{i,j} = \begin{cases} 1 & \text{for } i \in G_j \\ 0 & \text{for } i \notin G_j \end{cases}
\]
with \( q_j = |G_j| \). We will show that \( Q = Q_1, a(V_{N,k}, q_j) \).

From the construction of the \( MD(1,k) \) vote and quorum assignment that yields \( Q_1, a(V_{N,k}, q_j) \), it is trivially true that \( H \in Q \implies H \subseteq Q_1, a(V_{N,k}, q_j) \), i.e., \( Q \subseteq Q_1, a(V_{N,k}, q_j) \). (When \( H = G_j \), votes from nodes in \( H \) satisfy the quorum requirement in the \( j \)-th dimension and \( H \) is also tight.) Consider the \( j \)-th dimension of the MD vote assignment that is derived from the group \( G_j \in Q \). The set of groups \( C_j \) represented by this dimension is equal to \( \{ G_j \} \) and since \( Q_1, a(V_{N,k}, q_j) \) is the set of all groups that are proper subsets of \( V_{N,k} \), we have \( Q_1, a(V_{N,k}, q_j) \subseteq Q \). \( \Box \)

Lemma 3.1 guarantees that an \( MD(1,k) \) vote and quorum assignment can be found for any set of minimal groups \( Q \). In the constructive proof, since each group is represented by a separate dimension, the number of dimensions used is equal to \( |Q| \), which may be large. We present in what follows a technique that uses the fact that several groups may be representable by a single dimension of an MD vote and quorum assignment. Therefore, in practice, the number of dimensions needed to obtain an \( MD(1,k) \) vote assignment could be less than \( |Q| \).

3.3 Finding an SD vote assignment

In [3], a technique is described for testing if a set of groups \( Q \) is SD vote assignable. The following linear program, \( LP(Q) \), is setup using the groups in \( Q \),

\[
\text{Min: } \sum_{i=1}^{N} v_i + q
\]
\[
\text{s.t.: } \forall G \in Q : \quad \sum_{g \in G} v_g \geq q \quad (1)
\]
\[
\forall H \in \overline{\text{sup}}(Q|U) \cup \text{psub}(Q) : \quad \sum_{h \in H} v_h \leq q - 1 \quad (2)
\]
\[
v_i \geq 0, \quad i = 1, 2, \ldots, N
\]
\[
q \geq 1
\]

where,

- \( \overline{\text{sup}}(Q|U) \) is the set of all groups that are subsets of \( U \) and not supersets of any group in \( Q \), and
- \( \text{psub}(Q) \) is the set of all groups that are proper subsets of the groups in \( Q \).

If \( LP(Q) \) does not have a feasible solution then \( Q \) is not SD vote assignable, otherwise a feasible solution (which is rational) can be converted to an integer vote and quorum assignment.

Unlike [3], we are dealing here with sets \( Q \) with the minimality property. This allows us a further refinement of \( LP(Q) \) which is a result of the following lemma.

Lemma 3.2: Let \( Q \) be a set of groups satisfying the minimality property, i.e., \( \forall G, H \in Q : \quad G \nsubseteq H \). Then, \( \text{psub}(Q) \subseteq \overline{\text{sup}}(Q|U) \).

Proof: The proof is by contradiction.

Assume there is a group \( X \) that is in \( \text{psub}(Q) \) but not in \( \overline{\text{sup}}(Q|U) \). Since \( X \in \text{psub}(Q) \), \( X \) is a proper subset of some group \( A \in Q \). Furthermore, \( X \notin \overline{\text{sup}}(Q|U) \) so it is a superset of some (other) group \( B \) such that \( B \in Q \). However, then \( Q \) would violate the minimality property because the above facts imply that \( B \subseteq A \). This contradicts the premise. \( \Box \)

We can thus substitute constraint (2) in \( LP(Q) \) with,

\[
\forall H \in \overline{\text{sup}}(Q|U) : \quad \sum_{h \in H} v_h \leq q - 1 \quad (3)
\]

3.4 Algorithm for Finding MD(1,k)

Here we extend the procedure described in subsection 3.3 to find an \( MD(1,k) \) vote and quorum assignment for a set of groups \( Q \) satisfying the minimality property. Our algorithm (illustrated in Figure 1) finds an \( MD(1,k) \) vote assignment by testing to see if \( Q \) is SD vote assignable. If not, groups are systematically removed from \( Q \) until the groups that remain form an SD vote assignable set. The votes and quorum obtained from the solution form the assignment in the frst dimension. The set of groups removed from \( Q \) to make it SD vote assignable are then used as input to a second iteration to find the second dimension of vote and quorum assignment. This is repeated until no more groups remain. Since a set with one group is always vote assignable, in each iteration at least one group of \( Q \) is represented by the solution found and the algorithm is guaranteed to terminate. An efficient implementation of the algorithm is presented in [24]. Sections 4 and 5 contain examples of the application of our algorithm.

4 MD-Voting for Mutual Exclusion

The problem of mutual exclusion arises in many applications where a process must acquire exclusive access to
k := 0; Q := set of minimal groups; D := Δ;

Figure 1: Algorithm for finding MD(1,k) vote and quorum assignment

Figure 1: Algorithm for finding MD(1,k) vote and quorum assignment

...
Table 3: A tree-based coterie and its multidimensional vote assignment

The last write operation and two write operations are not executed concurrently. In general, synchronization of read and write operations can be ensured by requiring that each operation obtain permission of a group of nodes and the groups used by conflicting operations have non-empty intersection. Minimal groups of nodes that can allow a read and write operation to complete are called read and write groups respectively. The read and write coteries $R$ and $W$ are the sets of read and write groups used. ($W$ is a coterie and $R$ is its anti-coterie [25].) The synchronization requirements given above are satisfied if,

1. (Read/write intersection property) $\forall G \in R, H \in W: G \cap H \neq \phi$, and
2. (Write/write intersection property) $\forall G, H \in W: G \cap H \neq \phi$.

$R$ and $W$ have the minimality property and $W$ also has the intersection property. Since read operations can be executed concurrently, the set $R$ need not satisfy the intersection property, i.e., $R$ is in general not a coterie which is used for enforcing mutual exclusion. For a given $R$, to maximize read availability, $W$ equals the minimal set of minimal groups that have read/write and write intersection property. The set $W$, in general, is not unique for a given $R$. For example, let $R = \{\{1,2\}, \{3,4\}\}$, then the sets $\{\{1,3\}, \{1,4\}, \{2,3,4\}\}$ and $\{\{1,3\}, \{2,3\}, \{1,2,4\}\}$ can both be used as write coteries.

A replica control protocol corresponds to a read and a write coterie which satisfy the synchronization requirements. Thus, in the general case, consistency of replicated data is maintained using two possibly different sets of groups (one for reading and the other for writing) which have the minimality property. It is straightforward to see that an MD vote assignment can be obtained for each of them, one is used for reading and the other one is used for writing. In the general case, the MD vote assignments obtained for the read and write coteries may be different. Consequently, a node must use the appropriate MD vote assignment (based on the type of the request) to vote on each request. Thus, votes obtained for a read request cannot be used for a write request. Although it is feasible to implement read and write coteries with separate MD vote and quorum assignments, it is simpler and more efficient to allow votes obtained for reading to be augmented to a quorum for writing because transactions usually read the data before updating them. This will be similar to SD-voting where the same vote assignment is used to define both read and write coteries. In the next subsection, we describe a replica control protocol that uses a single MD vote assignment to define both read and write coteries and allows a read quorum to be augmented when the read data items are also updated.

### 5.1 A Replica Control Protocol Based on MD-Voting

In the design of a replica control protocol, an appropriate read coterie that provides high read performance is chosen and the corresponding write coterie is computed to satisfy the synchronization requirements. In general, the read coterie can be represented by an $MD(\ell, k)$ vote and read quorum assignment. Let $V_{N,k}R$ and $\Delta_2 = (r_1, r_2, \ldots, r_k)$ be the vote and read quorum assignment for an $MD(\ell, k)$-voting system used in reading and $Q_{\ell,k}(V_{N,k}, \Delta_2)$ represents the set of minimal groups defined by the assignment. To allow write operations to synchronize using the same vote and read quorum assignment, we define the write quorum $W_k = (w_1, w_2, \ldots, w_k)$ to be,

$$w_j = \sum_{i=1}^{N} v_{i,j} - r_j + 1, \quad \text{for } j = 1, 2, \ldots, k$$

We do not require that $w_j \geq \left\lceil \frac{\sum_{i=1}^{N} v_{i,j+1}}{2} \right\rceil$ votes, for $j = 1, 2, \ldots, k$. The write quorum $w_k$ will only ensure that groups that satisfy the write requirement intersect with all read groups of the $j^{th}$ dimension of the MD vote assignment. Since the read coterie is defined by $MD(\ell, k)$-voting, we must use $MD(k - \ell + 1, k)$-voting for writing to ensure that the read/write intersection property holds. Let $Q_{\ell+1-k}(V_{N,k}, \Delta_2)$ be the set of tight groups represented by the $MD(k + 1 - \ell, k)$ vote and write quorum assignment. The following lemma shows that $Q_{\ell,k}(V_{N,k}, \Delta_2)$ and $Q_{\ell+1-k}(V_{N,k}, \Delta_2)$ have the read/write intersection property.

**Lemma 5.1:**

$\forall G \in Q_{\ell,k}(V_{N,k}, \Delta_2), H \in Q_{\ell+1-k}(V_{N,k}, \Delta_2): G \cap H \neq \phi$

**Proof:**

Let $G$ and $H$ be two arbitrary groups in $Q_{\ell,k}(V_{N,k}, \Delta_2)$ and $Q_{\ell+1-k}(V_{N,k}, \Delta_2)$ respectively. Since $(\ell) + (k + 1 - \ell) > k$, there is some dimension $s$ such that,

$$\sum_{g \in G} v_{g,s} \geq \tau_s$$

and

$$\sum_{h \in H} v_{h,s} \geq w_s$$


\[Q = \{\{124\}, \{125\}, \{136\}, \{137\}, \{145\}, \{167\}, \{2346\}, \{2347\}, \{2356\}, \{2357\}, \{2467\}, \{2567\}, \{3456\}, \{3457\}, \{4567\}\} \]

\[V_{r,s} = \begin{pmatrix} 0 & 0 & 0 & 2 & 2 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 2 & 0 & 1 \\ 2 & 0 & 2 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}, \quad q_s = (6, 6, 6, 4, 4)\]
Since \( r + w > \sum_{i=1}^{N} v_{i,n} \), there must be a common node in \( G \) and \( H \) and hence \( G \cap H \neq \emptyset \).

Although the sets \( Q_{\ell,k}(V_{N,n}, E_k) \) and \( Q_{k+1, \ell-k}(V_{N,n}, E_k) \) have the intersection property which is necessary for read/write synchronization, \( Q_{k+1, \ell-k}(V_{N,n}, E_k) \) may not be a write coterie for \( Q_{\ell,k}(V_{N,n}, E_k) \) because it may not have the write/write intersection property which is required when version numbers are used. To achieve this, we can augment each group of \( Q_{k+1, \ell-k}(V_{N,n}, E_k) \) to include a group of \( Q_{\ell,k}(V_{N,n}, E_k) \). We define the write coterie \( W \) which is derived from \( Q_{\ell,k}(V_{N,n}, E_k) \) and \( Q_{k+1, \ell-k}(V_{N,n}, E_k) \) in the following way:

\[
W = \{ A \cup B | A \cup B \text{ is minimal and } A \in Q_{\ell,k}(V_{N,n}, E_k), \ B \in Q_{k+1, \ell-k}(V_{N,n}, E_k) \}
\]

It can be easily seen that when \( Q_{\ell,k}(V_{N,n}, E_k) \) and \( W \) are used for reading and writing, both the read/write and write/write intersection properties are satisfied. The latter property follows from the read/write intersection of groups in \( Q_{\ell,k}(V_{N,n}, E_k) \) and \( Q_{k+1, \ell-k}(V_{N,n}, E_k) \).

The protocol used is as follows: when reading, the operation obtains a read quorum in at least \( \ell \) dimensions and when writing, it must obtain a read quorum and a write quorum in \( \ell \) and \( k+1-\ell \) dimensions, respectively. If the writing of the data is followed by its read (which is the typical case), the write operation only needs to obtain a write quorum and the method thus allows the read quorum to be augmented.

A special case of the protocol is when MD(1,k)-voting is used for reading. Then, we can use the method in Section 3 to find an MD(1,k)-voting read quorum assignment for the read coterie. The corresponding write coterie will be represented by using an MD(k,k)-voting write quorum assignment. In this case, the read operation can proceed if it can obtain a read quorum in any one dimension. If the transaction wishes to update the data after reading it, the vote received for the read request must be supplemented with additional votes such that in each dimension the number of votes received is greater than or equal to the write quorum for that dimension. The general case where MD(\ell,k)-voting (arbitrary \( \ell \)) is used, is more difficult as we do not yet have an algorithm to find an MD(\ell,k) assignment when \( \ell \neq 1 \). However, if the read coterie is derived from some logical structure, such as the example described in the next subsection, we may be able to use the structure to formulate an MD assignment.

5.2 Example

The structured coterie concept can also be applied to synchronize reading and writing of replicated data. Similar to mutual exclusion, the resulting read and write coteries may not be SD-vote assignable. Structured read and write coteries can achieve higher load sharing than SD-voting which results in lower response times. Also, the number of messages used in the execution of each operation is reduced.

The replica control method presented in [21] organizes the nodes of the system into a logical grid consisting of \( m \) rows and \( n \) columns. The read coterie consists of groups of \( m \) nodes where one node is selected from each column and a write group consists of nodes in a read group and all nodes in a column of the grid (the coteries used in this method are generally not SD-vote assignable). The simulation study in [21] showed that the response times of transactions in systems using the grid protocol are significantly lower than those that use SD-voting for the same number of nodes. Also, an increase in the number of nodes in a system using SD-voting will not result in much reduction in response time because the load is not shared effectively. Systems using the grid protocol have higher maximum throughput and lower response time.

In [21] we have used coteries to implement the grid protocol. Each operation knows the topology and the position of the nodes in the grid. An operation checks whether the collection of responses constitutes a group that can permit it to proceed. The read and write coteries used in the grid protocol can be represented using MD-voting. In fact, a single MD vote assignment can be used to represent both coteries. The MD vote assignment used for an \( m \times n \) grid network consists of \( m \) dimensions and a node \( v \) has \( v_{i,j} = 1 \) if it is in column \( j \), otherwise \( v_{i,j} = 0 \). The read and write quorums used are \( r_{\ell} = (1,1,\ldots,1) \) and \( w_{\ell} = (m,m,\ldots,m) \), respectively and MD(\( m, n \))-voting is used for reading and writing, respectively. Using the MD-voting implementation of the grid protocol, operations do not need to know the topology of the grid. The read coterie \( Q_{\ell,n}(V_{N,n}, E_k) \) consist of groups of nodes with exactly one node from each column and the set \( Q_{\ell,n}(V_{N,n}, E_k) \) consists of groups of all nodes in a column of the grid. The general case where MD(\( \ell, k \))-voting (arbitrary \( \ell \)) is used, is more difficult as we do not yet have an algorithm to find an MD(\( \ell, k \)) assignment when \( \ell \neq 1 \). However, if the read coterie is derived from some logical structure, such as the example described in the next subsection, we may be able to use the structure to formulate an MD assignment.

6 Conclusion

In this paper, we have introduced the concept of a multi-dimensional vote and quorum assignment which is a generalization of standard voting. In multi-dimensional voting, the vote assigned to a node and the quorum assignment are vectors of non-negative integers and each dimension is similar to standard voting. We have shown that any set of minimal groups can be represented by multi-dimensional voting. We have also shown that it is more general than standard voting and is as powerful as the coterie concept, which is the general approach for achieving mutual ex-
finding a multi-dimensional vote and quorum assignment has the advantage that it is flexible and can be easily implemented. We have developed an efficient algorithm for finding a multi-dimensional vote and quorum assignment for any set of minimal groups and its use was shown by finding multi-dimensional vote assignments for some non SD-vote assignable coteries. We also discuss the use of multi-dimensional voting to synchronize reading and writing of replicated data and showed an example of its use to represent the read and write coteries of a replica control protocol based on a logical grid network.

Figure 2: A Grid Network

\[
V_{4,3} = \begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 0 & 0 \\
1 & 0 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1 \\
\end{pmatrix},
Z_3 = (1, 1, 1),
W_3 = (2, 2, 2)
\]

Table 4: Multi-dimensional vote and quorum assignment for the 3x2 grid system

References