INFORMATION SCIENCES **74,** 223–246 (1993)

A New Transformation Method for Nondominated Coterie Design

DAVID SHOU

and

SHENG-DE WANG

Department of Electrical Engineering,
National Taiwan University,
Taipei 106, Taiwan, Republic of China

Communicated by Abraham Kandel

ABSTRACT

Coteries are general schemes for distributed mutual exclusion solutions. Garcia-Molina and Barbara developed a transformation algorithm to transform one nondominated (ND) coterie to another [5]. The algorithm may properly be used to partially enumerate ND coteries. However, the transformation algorithm proposed is not appropriate for the case of large coterie design because properties of the new produced ND coterie are not easily predictable. We introduce a hierarchical transformation concept to provide a simple way to produce new larger ND coteries through transformation from smaller ND coteries. By systematically applying our transformation, coteries can scale well for a large number of nodes. Furthermore, the new generated coteries will inherit the properties of original coteries. This makes the transformation more predictable than trial and error. We also devise a heuristic algorithm to construct hierarchical structures aimed at maximizing availability in the context of heterogeneous systems.

1. INTRODUCTION

The distributed mutual exclusion problem is about how to guarantee mutually exclusive access to a critical section among a number of processes located in different nodes in a distributed system. It is crucial in the design of distributed systems. For example, replication increases data availability

Correspondence to S.-D. Wang.

*This research was partially supported by grant NSC82-0408-E-002-024 from the National Science Council, Taiwan, ROC.

655 Avenue of the Americas, New York, NY 10010 0020-0255/93/$6.00

and system reliability; however, two write operations or the read and write operations performed on a replicated data object by various concurrent transactions located in different nodes must be mutually exclusively executed. That is, certain protocols must be imposed to enforce serialization and prevent time-dependent errors.

A number of methods have been proposed to solve the distributed mutual exclusion problem [9, 18, 6, 16, 5, 15, 13]. They can be classified into two categories: quorum-based methods (permission-based methods) and token-based methods. Token-based methods rely on token transfer protocols to ensure liveness and the fact that only the token holder can execute critical operation to achieve mutual exclusion. Quorum-based methods rely on enough rights collection to permit only one critical section executed at a time. However, token-based methods suffer from poor failure resiliency. That is, if the node holding the token fails, a complex token regeneration protocol, which is not trivial, must be executed [12]. Therefore, considering fault tolerance, the quorum-based schemes are thought to be good approaches to achieve mutual exclusion in, especially large, distributed systems. The general notation of quorum scheme is coteries which may be implemented by multidimensional voting [4].

The coterie design is not a simple problem. Vote assignment, a subproblem of coterie design, has been intensively studied [5, 6, 7]. Only heuristic algorithms are available. Toward the general coterie design, Garcia-Molina and Barbara defined a domination relation between coteries in order to narrow down the solution space considered by the system designer. It can be shown that once one coterie dominates another, the dominating one compares favorably with the other to be adopted as a solution for the distributed mutual exclusion problem because the dominating one incurs lower communication cost and gains higher fault-tolerant capability. We say a coterie is dominated iff there is at least one other coterie which dominates it. If no such coterie exists, then the coterie is nondominated (ND). A dominated coterie should not be used because the dominating coterie provides strictly greater access capability while still preserving mutual exclusion.

Checking domination of coteries seems to be a hard problem. Therefore, Garcia-Molina and Barbara [5] developed a transformation algorithm (TA) to enumerate ND coteries for small systems. By this enumeration, some objective functions may be maximized. However, the TA proposed in [5] is not appropriate for the case of large coterie design due to the unexpected properties of the new produced ND coterie. The partial enumeration algorithm proposed in [5] is also not appropriate due to its high computation complexities. Recently, several coterie designs have been reported [11, 1, 8]. Maekawa [11] proposed a method (finite projective planes) for generating equal-sized quorums, which involve communication

with $\sqrt{n}$ nodes, to guarantee mutual exclusion. Agrawal and El Abbadi [1] proposed a fault-tolerant tree quorum algorithm that requires $O(\log n)$ messages in the best case. This is achieved by imposing a logical tree structure on the network, and thus reduces the quorum size significantly. The hierarchical quorum consensus algorithm proposed by Kumar [8] is a fully distributed algorithm wit $O(n^{0.63})$ complexity. But these works did not prove the ND property of their coteries.

Our objective is to provide a simple way to produce new larger ND coteries through transformation from smaller ND coteries. By systematically applying our transformation, coteries can scale well for a large number of nodes. Furthermore, the new generated coteries will inherit the properties of original coteries. This makes the transformation more predictable than the approach of trial and error. This transformation is based on the concept of hierarchy; by means of repetitive or recursive transformation, a large coterie with predictable features can be obtained. The coteries can be used to solve the distributed mutual exclusion problem. In the context of heterogeneous systems, we also devise a heuristic algorithm to construct hierarchical structures from which coteries for maximizing availability can be derived.

The organization of this paper is as follows. In Section 2, we present the HT, the hierarchical transformation. Then, we focus our discussion on regular structures as defined in Section 3. Moreover, some famous quorum schemes can easily be derived from systematically applying our transformation. A heuristic algorithm aimed at maximizing the availability to construct dedicated hierarchical structures based on a heterogeneous system is presented in Section 4. We conclude the paper with a summary of our results.

2. HT: HIERARCHICAL TRANSFORMATION

In this section, we review the concept of coteries, and then propose a transformation method, called the hierarchical transformation (HT), based on the concepts of coteries and hierarchy. The algorithm takes a known ND coterie as the original coterie to construct larger ND coteries using some primitive ND coterie. Good candidates for primitive coteries and original coteries can be defined by majority voting for a set of an odd number of nodes, small ND coteries with nodes within five enumerated by exhausted searching, or even the ND coteries obtained by HT; any specific ND which can easily be derived will do.

Since our approach is closely related to the concept of coteries [5], it is helpful for us to review the definition of coteries. Let U be the set of nodes that compose the system. A coterie, S, under U is a set of sets

where each set in S is called a quorum. The following conditions hold for all quorums in a coterie S:

(i) $G \in S$ implies that $G \neq \emptyset$, and $G \subseteq U$.

(ii) (Intersection Property) If G and H are quorums in S, G and H must have a nonempty intersection, i.e., $G \cap H \neq \emptyset$.

(iii) (Minimality Property) No two quorums, G and H, exist in S so that G is a subset of H.

By their intersection property, the members in a coterie can be used as quorums to guarantee mutual exclusion in a distributed system. The minimality property is not necessary for correctness, but can enhance efficiency.

The concepts of coteries have been studied at length by Garcia-Molina and Barbara [5], where the domination for coteries has also been defined. Let R, S be coteries. R dominates S *iff* $R \neq S$ and, for each $H \in S$, there is a $G \in R$ such that $G \subseteq H$. It can be shown that R compares favorably with S to be adopted as a solution for the mutual exclusion problem because R incurs lower communication cost and gains higher fault-tolerant capability. We say a coterie S is dominated *iff* there is another coterie which dominates S. If no such coterie exists, then S is nondominated (ND). A dominated coterie should not be used because the dominating coterie provides strictly greater access capability while still preserving mutual exclusion.

There are many facets of mutual exclusion solutions for distributed systems: the availability or reliability, the communication costs, the response times, etc. If we search for solutions that attain the maximum in reliability, the voting ND coteries should be considered because they have been proven to be optimal for homogeneous systems. If the communication costs are the primary concern, the primary site approach performs the best. Practically, something that falls between these two extremes is desirable. This is why ND transformation is important.

The process of hierarchical transformation (HT) is stated in detail.

DEFINITION 1. (HT) Given an original coterie S under U_S and a primitive coterie R under $U_R, |U_R| > 2$ and $U_s \cap U_R = \emptyset$, a member x in U_S is chosen. We define HT as

$$S' = HT(S, x, R) = S_{x'} \cup E_x \text{ under } U_{S'} = (U_s - x) \cup U_R,$$

where

$$S_x = \{Q | x \in Q, Q \in S\},$$
$$S_{x'} = S - S_x,$$
$$E_x = \{H | H = (Q - \{x\}) \cup G, Q \in S_{x'}, G \in R\}. \qquad \square$$

For example, let $S=\{\{1,2\},\{1,3\},\{2,3\}\}$, $x=1$, $R=\{\{4,5\},\{4,6\},\{5,6\}\}$.

$$S'=HT(S,x,R)=\{\{2,3\},\{4,5,2\},\{5,6,2\},\{4,6,2\},$$
$$\{4,5,3\},\{5,6,3\},\{4,6,3\}\}$$

where

$$S_x=\{\{1,2\},\{1,3\}\},$$

$$S_{x'}=\{\{2,3\}\},$$

$$E_x=\{\{4,5,2\},\{5,6,2\},\{4,6,2\},\{4,5,3\},\{5,6,3\},\{4,6,3\}\},$$

$$U_S=\{1,2,3\},$$

$$U_R=\{4,5,6\},$$

$$U_{S'}=\{2,3,4,5,6\}.$$

The node x in system U_S selected in the transformation can be interpreted as a virtual node corresponding to a subsystem U_R. Because only one quorum in U_R will be established at a time, the quorum is of authority to give its (virtual node associated with U_R) permission to the node requesting critical operation when a quorum is constructed in U_R. The critical operation may be committed only if "enough" permissions (a quorum, no matter whether formed by real nodes and/or virtual nodes, over the original coterie) are collected. That is, a quorum in U_S is established in which the virtual node, such as x in Definition 1, is substituted by a quorum in U_R. Note that the identifiers are not important since renaming can easily be applied.

Now the proof of the correctness of the HT for the coterie transformation is given.

THEOREM 1. *The set of quorums generated by applying coteries to HT meets the conditions of the intersection and the minimality properties. That is, it is a coterie.*

Proof. First, we prove that the quorums transformed by the HT meet the requirement of intersection. Let two quorums Q_1 and $Q_2 \in S'$ under $U_{S'}$ where S' is derived from HT. By Definition 1, it is clear that $S_{x'} \cap E_x = \emptyset$, $S' = S_{x'} \cup E_x$. There are three cases to be examined.

Case 1: Q_1 and $Q_2 \in S_{x'}$. Since $S \supset S_{x'}$ and S is a coterie, $Q_1 \cap Q_2 \neq \emptyset$.

Case 2: Q_1 and $Q_2 \in E_x$. By Definition 1, there must exist Q_{1S}, $Q_{2S} \in R$ such that $Q_1 \supset Q_{1S}$, $Q_2 \supset Q_{2S}$, where R is a coterie. Since $Q_{1S} \cap Q_{2s} \neq \emptyset$, $Q_1 \cap Q_2 \neq \emptyset$.

Case 3: $Q_1 \in S_{x'}$, $Q_2 \in E_x$. Because the set $(Q - \{x\})$ will intersect any $Q_1 \in S_{x'} (S_{x'} = S - \{Q | x \in Q,\ Q \in S\})$ and $Q_2 \supset (Q - \{x\})$ where $Q \in Sx$, $Q_1 \cap Q_2 \neq \emptyset$.

Hence, any pair of the quorums transformed by HT has a nonempty intersection.

Next, we show that the coterie transformed by HT meets the requirement of minimality.

Case 1: Q_1 and $Q_2 \in S_{x'}$. Since $S \supset S_{x'}$ and S is a coterie, $Q_1 \not\subset Q_2$.

Case 2: Q_1 and $Q_2 \in E_x$. By Definition 1, there must exist $Q_{1S}, Q_{2S} \in R$ such that $Q_1 = (H_1 - \{x\}) \cup Q_{1S}$, $Q_2 = (H_2 - \{x\}) \cup Q_{2S}$, $H_1, H_2 \in S_x$, $S \supset S_x$, where S, R are coteries. Since $U_S \cap U_R = \emptyset$ and $Q_{1S} \not\subset Q_{2S}, Q_1 \not\subset Q_2$.

Case 3: $Q_1 \in S_{x'}$, $Q_2 \in E_x$. $Q_1 \not\subset Q$ (by minimality property over U_S) means $Q_1 \not\subset (Q - \{x\})$ where $Q \neq Q_1$, $Q \in S_x$, $Q \in S$. $Q_1 = (Q - \{x\}) \cup G$ where $G \in R$ and $U_S \cap U_R = \emptyset$. Therefore, $Q_1 \not\subset Q_2$. On the other hand, $Q_2 \cap U_R \neq \emptyset$, but $Q_1 \cap U_R = \emptyset$. It leads to $Q_2 \not\subset Q_1$. That is, Q_1 has something that Q_2 does not have and vice versa.

Based on the discussion above, we conclude that the S' derived from HT is a coterie because it meets the nonempty intersection and minimality properties. □

Furthermore, we prove that coteries obtained by HT with both original and primitive coteries being ND are ND.

THEOREM 2. *The coteries transformed by HT with both original and primitive coteries being ND are ND.*

To prove Theorem 2, we first provide the following lemma which is adapted from [5]. The lemma indicates that if no quorum G exists for either conditions, the coterie S is ND. One condition is that some quorum can be reduced and still satisfy the nonempty intersection property. The other condition is that other quorums (intersecting with all quorums defined) exist other than those defined quorums and their supersets.

LEMMA 1. *A coterie S under U is dominated iff there exists a quorum $G \subseteq U$ such that for all $H \in S$, $G \notin S$, $G \cap H \neq \emptyset$, and at least one of the following conditions holds.*

(i) *$R = S \cup \{G\}$ is a coterie.*

(ii) *There are one or more $H_1, H_2, H_3, \ldots, H_n \in S$ such that $G \subset H_1, H_2, H_3, \ldots, H_n$. Then $R = (S - H_1 - H_2 - H_3 - \cdots - H_n) \cup \{G\}$ is a coterie.*

Proof. Consider condition (i). It is obvious that there exists an $H_i' = H_i \subseteq H_i$ for all $H_i \in S$ and $H_i' \in R$. Because $S \neq R, S$ is dominated by R. Consider condition (ii). We see that $S \neq R$, and there always exists an $H_i' \subseteq H_i$ for all $H_i \in S$ and $H_i' \in R$. Therefore, S is dominated by R. If S is dominated, there exists R dominating S, that is, $R \neq S$ and, for each $H \in S$, there is a $G \in R$ such that $G \subseteq H$. Two conditions should be considered for $R \neq S$: $S \subset R$ and $S \not\subset R$. $S \subset R$ implies that there exists $G \in (R - S)$; condition (i) holds. $S \not\subset R$ means that $S - (S \cap R) \neq \emptyset$ and $R - (S \cap R) \neq \emptyset$(if $R - (S \cap R) = \emptyset$, $R \subset S$, which is contradictory to R dominates S). Let $H_1 \in (S - (S \cap R))$. There must exist $G \in (R - (S \cap R))$ such that $G \subset H_1$. The two conditions in the lemma are necessary and sufficient for dominated coteries. □

Now, if we can prove no such set G satisfies either of the two conditions above, the coteries S obtained from HT are ND.

PROOF OF THEOREM 2. Condition (ii) is impossible. Assume that G exists. There is a B such that $G = H - B$, $H \in S$, and $B \neq \emptyset$. Let $y \in B$, either $y \in U_S$, or $y \in U_R$ exclusively. Consider the condition that $y \in U_R$. Then $G \cap U_R$ cannot intersect with all $H_R \in R$ where R is the ND coterie associated with U_R. That is, not all H which include H_R will intersect G. This leads to a contradiction. Now consider the other condition that $y \in (U_S - \{x\})$ as HT(S, x, R) is applied. Then $G \cap U_S$ cannot intersect with all $H_S \in Sx'$ where $Sx' = S - \{Q | x \in Q, Q \in S\}$, and S is the ND coterie associated with U_S. Therefore, a G such that $G \subset H$ and $G \cap H \neq \emptyset$ does not exist.

Condition (i) is impossible: $G \not\subset H$ means that $G - (G \cap H) \neq \emptyset$ and $H - (G \cap H) \neq \emptyset$. Let $A = G - (G \cap H)$ and $B = H - (G \cap H)$. Let $y \in A$, either $y \in U_S$, or $y \in U_R$ exclusively. Consider the condition that $y \in U_R$. Any set $H \in S'$ generated from HT which intersects G will remain unchanged if G becomes $G - y$. It is because $G \cap U_R$ must be a superset of some $H_R \in R$ because R is the ND coterie associated with U_R. Now consider the other condition that $y \in (U_S - \{x\})$ as HT(S, x, R) is applied. Similarly, $G \cap (U_S - \{x\})$ must be a superset of some $H_{Sx'} \in Sx'$ where $Sx' = S - \{Q | x \in Q, Q \in S\}$ because S is the ND coterie associated with U_R. So the intersection property remains the same for reducing G to $G - A$.

Also, a similar discussion in condition (ii) applying to $G-A$ leads to the conclusion that no $G-A$ as well as G exists for condition (i). That is, the coteries derived from HT with original and primitive coteries being ND are ND. □

The same transformation may be repeatedly applied to the new generated coterie so that the universe may expand without limit. Also, the coterie derived from HT corresponds to a logical hierarchical structure. To illustrate the concept by an example, a system of 12 real nodes numbered top-down and left-to-right in a distributed system are logically organized into a hierarchical structure as shown in Figure 1. A circle denoting a real node corresponds to a node of the system, while a disk denoting a virtual node corresponds to a subhierarchy with several nodes. Primitive coteries have been defined for each group (a small universe, a system not expanding the virtual nodes yet) as shown in the set notation in the top level and each subhierarchy. By the coterie obtained from repetitive HT according to the structure with those primitive coteries shown, we can show that $\{1,5,6\}$ is a quorum for the overall system because a quorum, G, can be derived as follows:

$$G \rightarrow \{A\colon 1, C\} \rightarrow \{A\colon 1,\ A\colon 5,\ B\colon 6\}$$

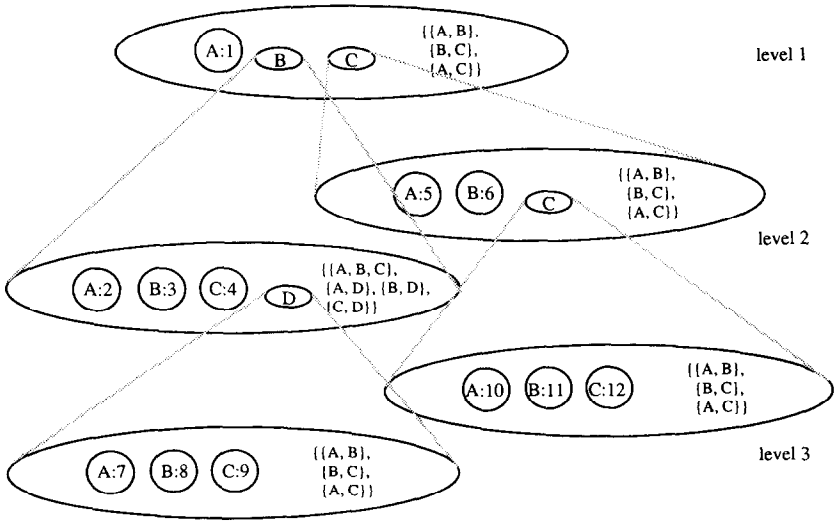


Fig. 1. Repeat hierarchical transformation.

where the symbol "→" denotes one-step derivation and C has been replaced by quorum {A: 5, B:6} defined in level 2. As can be seen from Figure 1, in level 1, A is a real node and has corresponded to node 1, as denoted by A: 1, and C is a virtual node, which in turn is associated with a group defined in level 2. The other quorums can be derived by a similar procedure, which can be referred to as top-down derivation as in the derivation of sentences from their grammar rules.

Quorum invocation algorithms based on coteries obtained by HT have many possibilities. By adapting from the algorithm given in [8], we give one general algorithm for constructing a quorum based on the structures corresponding to hierarchical coteries.

Algorithm 1 (A general algorithm for constructing a quorum based on the hierarchical structure)

```
GetQuorum(start){
if((Assemble(start) == 1)
      return(yes); /* request successful */
      else{
      unlock all nodes it locked;
      return(no); /* request unsuccessful */
        }
}
Assemble(virtual_node){
      for all node in the group corresponding to virtual_node do {
      if (node is real node)
            {if (node is unlock)
                  {lock node;
                  lock_nodes = lock_nodes ∪ node;}}
            else /* virtual node */
            if Assemble(node)
            lock_nodes = lock_nodes ∪ node;
      if (lock_nodes is a primitive quorum defined in the group
      corresponding to virtual_node)
      return(1);
      }
      return(0);
}
```

The main section of the algorithm makes a call to the recursive function **Assemble** to form a quorum. The main section returns yes if it is successful in assembling a valid quorum in the only group, corresponding to the virtual node *start*, at the top level; else it releases all nodes it locks and returns no. The function **Assemble** checks all nodes in group *start*. If it is an unlocked real node, the algorithm locks the node and includes it to the set *lock_nodes*. If it is a virtual node, the **Assemble** calls itself recursively. If the **Assemble** returns 1 for a virtual node, the node will also be included in the set *lock_nodes*. The recursive procedure repeats until a quorum is constituted or the condition is met that further requesting is of no use to reach quorum consensus.

The quorum-scheme-based hierarchical transformation described above is very general both in the definition of the logical structure and in the choice of primitive coteries. Sometimes, we are only interested in regular structures where the groups are associated with certain primitive coteries for their simplicity, easy implementation, and evaluation. Given a primitive coterie for each group in the hierarchy and a rule for constructing the hierarchical structures, we may specify a quorum scheme (a class of coterie). Note that the coterie defined by the HT can serve as a primitive coterie again for a group in a larger hierarchy. Due to this recursive feature, the quorum schemes defined by the HT are scalable well for a large-scale system. We will discuss a simple rule to construct regular hierarchical structures in the following section. Associating each group of the logical hierarchy with a certain primitive coterie, it leads to various quorum schemes with different properties.

3. REGULAR HIERARCHICAL QUORUM SCHEMES (RHQS)

For simplicity and clarity, we illustrate the HT application in a more restricted manner. A class of hierarchical quorum paradigms with regular structure and a uniform primitive coterie will be considered. A hierarchical quorum scheme is said to be regular if: (1) the logical structures are regularly constructed using the same rule for each expansion (how many and which node(s) should be virtual node(s)), and (2) all of the groups in the logical structure uniformly adopted the same definition of their primitive coteries. With these two conditions, a scalable quorum scheme can be defined. In this section, we will first define the regular structure, and then we use a voting mechanism, one important class of coterie, to serve as the primitive coterie for each group in the hierarchy. We will show that the Tree quorum [1] and the Hierarchical Quorum Consensus [8] can be easily derived from this paradigm.

3.1. REGULAR HIERARCHICAL STRUCTURES

A hierarchical structure can be constructed starting from a given virtual node, which can be considered to represent a hierarchy itself. A virtual node is then expanded with a group of nodes, which again consists of a number of virtual nodes and a number of real nodes. The following definition describes the regular hierarchical structure for the HT.

DEFINITION 2. A hierarchical structure for the HT is said to be regular if

(1) each group contains the same numbers of virtual nodes and real nodes,
(2) each virtual node except the ones at the last level is expanded with a group of nodes,
(3) each real node at all levels and each node in the groups at the least level correspond to a real node of the underlying system. □

It can be seen from Definition 2 that a regular hierarchical structure for using in the HT can be characterized by two parameters: the total number of nodes and the number of virtual nodes in a group. Note that each virtual node will be associated with one HT.

DEFINITION 3. Let RH(B, E) be a regular hierarchical structure, where B denotes the branch degree (or the number of nodes in each group) and E is the expansion degree (or the number of virtual nodes in each group), $0 \leqslant E \leqslant B$. □

For example, Figure 2 shows all instances of RH(B, E) with branch degree three, $B = 3$, and three levels. In one extreme, the one with zero expansion degree degenerates into no structure at all. In the other extreme, the structure with expansion degree three happens to be the hierarchical quorum structure proposed by Kumar [8]. Let $H(O)$ be the logical hierarchical structure proposed in [8], where O denotes the out-degree of each internal node in the hierarchy. Thus, the hierarchical structure $H(O)$ is a special case of RH(B, E) since the regular structure RH(B, E) becomes exactly the $H(O)$ structure, where both B and E are equal to O.

3.2. GHV: GENERAL HIERARCHICAL VOTING

To clarify the applications of HT, we describe a hierarchical quorum scheme subjected to the use of voting as the primitive coterie for each

group, and we call it general hierarchical voting (GHV). We will modify Algorithm 1 for use in GHV, and show that two existing algorithms are instances of a quorum scheme of the GHV. A new algorithm with desirable properties devised from this paradigm will also be discussed.

The Logical Structure and Quorum Definition

Developing a GHV method can be split into two phases: the logical structured quorum definition phase and the quorum constitution phase.

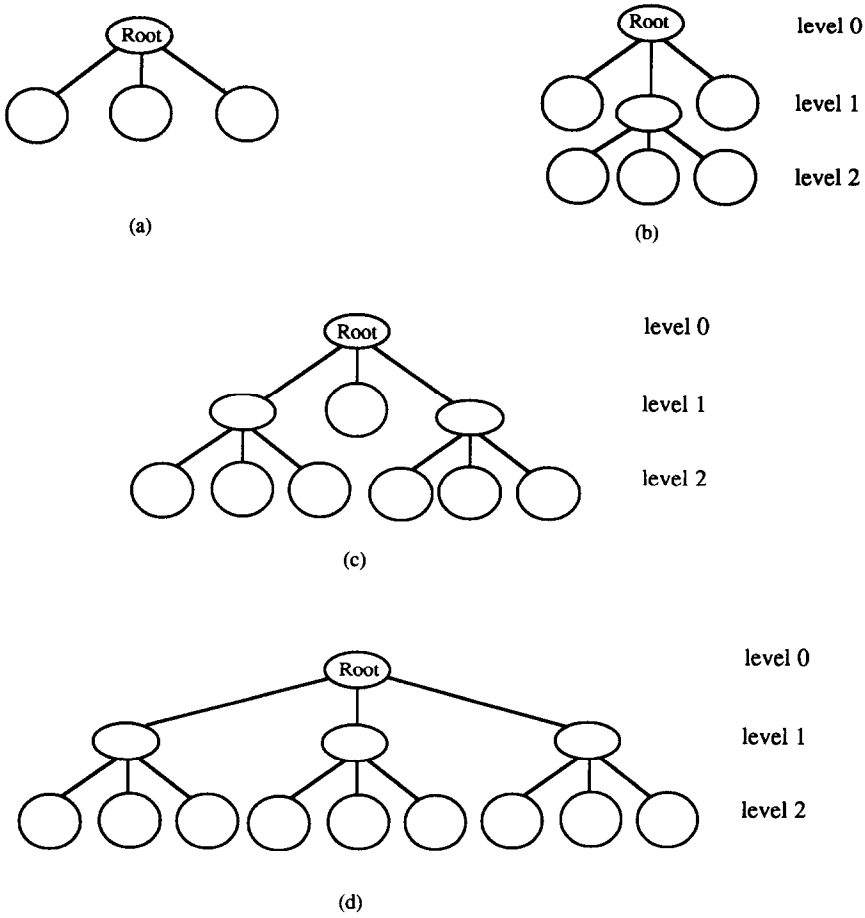


Fig. 2. A class of regular hierarchical structures with three levels: (a) RH(3,0), (b) RH(3,1), (c) RH(3,2), (d) RH(3,3).

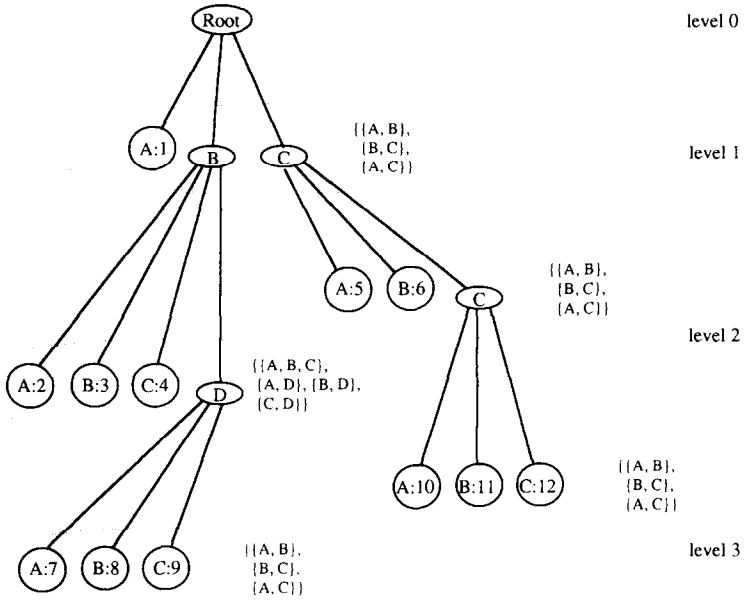


Fig. 3. The hierarchical structure corresponding to Figure 1.

Given a set of n real nodes in a system, we first logically organize the nodes into a hierarchical structure as depicted in Figure 3. The coterie is then defined while assuming HT is repeatedly being applied based on the structure. Whenever a real node in the system requests access to the critical section, it invokes a quorum constitution algorithm. Note that, if an instance of GHV is based on a regular logical structure, the definition phase can be merged into the constitution algorithm. That is, no explicit definition for a coterie is required since the coterie is implicitly defined by HT, the logical regular structure, and the voting quorum scheme.

In GHV, the primitive coterie for each group in the logical structure is defined by a form of voting. In a group, a primitive quorum is achieved if the number of nodes giving the permissions exceeds the required votes. An overall quorum is achieved if a primitive quorum at the top level is achieved. For example, the primitive coteries for all of the groups shown in Figure 3 are subject to majority voting, which is a form of quorum consensus, except that of the group corresponding to virtual node B in

level 1. So it should become $\{\{A,B,C\},\{A,B,D\},\{B,C,D\},\{A,C,D\}\}$ to fit the majority rule which is not ND.

Quorum Construction

The quorum construction algorithm as given in Algorithm 1 can be simplified for the use of GHV by means of replacing the primitive quorum testing by a counting of votes. The details of the quorum constitution algorithm for GHV are presented in the virtual code shown in Algorithm 2, and a brief description follows. In this algorithm, we use voting to define the primitive coterie which then defines quorums for those groups corresponding to virtual nodes.

Algorithm 2 (GHV quorum construction)

```
GetQuorum(start){
if ((Assemble(start) == 1)
        return(yes); /* request successful */
        else{
        unlock all nodes it locked;
        return(no); /* request unsuccessful */
          }
}

Assemble(node){/* virtual node */
        while (node -> quorum - num_locked ≤ tot_num -
        num_examined){
        num_examined = num_examined + 1;
        if (node is real)
                {if (node is unlock)
                        {lock node;
                        num_locked = num_locked + 1;}}
                else /* virtual node */
                if Assemble(node)
                num_locked = num_locked + 1;
        if (num_locked < node -> quorum)
        return(1);
        }
        return(0);
}
```

The data structure *node* used in Algorithm 2 is illustrated below.

```
struct node{
        boolean real;
        int tot_num;
        int quorum;
        list of *node;
        }
```

Each node records the following information: 1) a flag, *real*, denoting whether or not it is a real node; if not, the following data are required as well; 2) *tot_num* denoting the number of its children; 3) *quorum* denoting the number of votes needed; and 4) a list of pointers indicating its children. The variable *num_examined* refers to the number of children examined, and *num_locked* the number of children actually locked at the request of this algorithm.

The main section of the algorithm makes a call to the recursive function **Assemble** to form a quorum. It returns to 1 if it is successful in assembling the desired quorum for the group corresponding to the virtual node; else it returns 0. If the node is a real node, the algorithm checks if the node has been locked; if so, it just increases *num_examined* by 1; else it locks the node and returns 1 and increases *num_examined*. If it is a virtual node, then **Assemble** is called recursively. It repeats until a quorum is constituted or the condition is met that further requesting is of no use to win quorum consensus. The early giving up condition can improve the response time as compared with the algorithm in [8].

The algorithm is a contention-based one; that is, once it does not form a quorum, it releases all of the nodes it has locked. This resolves the deadlock problem intrinsically. Other quorum construction algorithms also exist. For instance, the timestamp-based approach resolves conflicts by a timestamp. It can be free from deadlocks and starvation [11].

Tree Quorums Equivalence

The binary tree quorums are equivalent to those derived from a GHV scheme with a logical structure RH(3,2) and the primitive coterie defined by majority voting. Let us compare the tree quorums method [1] with the GHV scheme corresponding to the RH(3,2) structure. The tree quorum algorithm constructs a quorum by one of the following three cases:

1) {root} ∪ {members from the quorum set of the left subtree},
2) {root} ∪ {members from the quorum set of the right subtree},

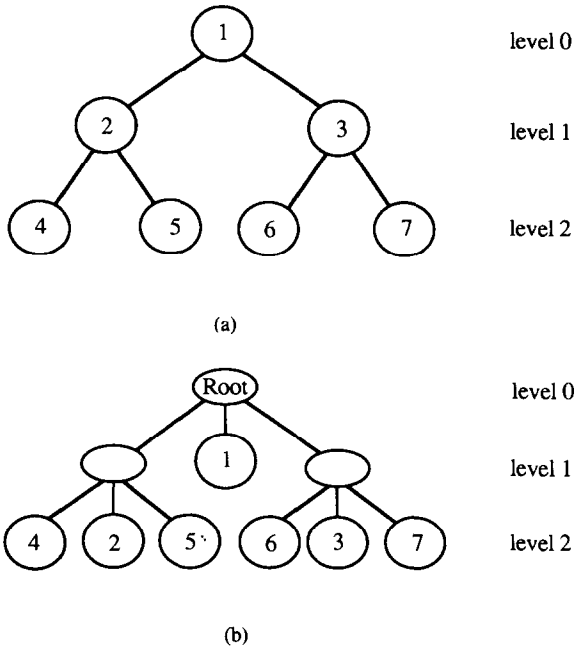


Fig. 4. Equivalent quorum structures: (a) tree quorum, (b) GHV quorum.

3) {members from the quorum set of the left subtree} ∪ {members from the quorum set of the right subtree}.

A quorum which can be derived by the GHV algorithm based on the RH(3,2) logical structure is the majority of votes of the top level, which consists of three nodes; one real node and two virtual nodes. Thus, although a simple mapping, it is easy to see that the binary tree quorums are equivalent to those of the GHV scheme based on RH(3,2).

An example is given in Figure 4 to illustrate the equivalence relation. For a system with seven nodes, the sets of quorums for both schemes are {{1, 2, 4}, {1, 2, 5}, {1, 3, 6}, {1, 3, 7}, {1, 4, 5}, {1, 6, 7}, {2, 3, 4, 6}, {2, 3, 4, 7}, {2,3,5,6}, {2,3,5,7}, {2,4,6,7}, {2,5,6,7}, {3,4,5,7}, {4,5,6,7}}. The original tree algorithm [1] prefers the quorums with smaller size. It can also be done in Algorithm 2 by imposing a selection mechanism that prefers real nodes over the virtual nodes. This preference makes the expected quorum size as small as possible.

It can also be shown that the quorums constructed from the GHV scheme with the regular structure RH(3,3) are the same as those con-

structed by HQC proposed by Kumar [8]. Hence, we have proven that the tree quorum and HQC are both ND. The quorums constructed by the GHV scheme with regular structure RH(3,1) are a new set of quorums which are referred to as level quorums. A salient feature pertaining to RH(3,1) is that there is exactly one group for each level, and each group contains two real nodes. The level quorums can be implemented by single-level weighted voting, whereas the HQC and the tree quorum cannot in general.

Systematically applying HT will result in large ND coteries whose properties are expectable. HQC, a fully distributed solution, incurs more communication cost ($O(n^{0.63})$, but can achieve higher availability. The tree quorum scheme incurs less communication cost ($O(\log n)$) in the best case and degrades gracefully, but sacrifices some availability as compared to HQC. These properties can be expected when we systematically apply the HT transformation. Therefore, by systematically applying the HT transformation concept, with selected parameters (original and primitive coteries and a structure), the system designer may derive some large expectable coteries to resolve the mutual exclusion problem in distributed systems.

4. STRUCTURES FOR HETEROGENEOUS SYSTEM

Fault tolerances are crucial, especially in distributed systems. Nodes may fail. Links may be broken. Messages may be lost. The mutual exclusion mechanism should survive under the vulnerable environment. To maximize the reliability of a distributed system in general is difficult. Garcia-Molina and Barbara [2, 17] have shown that the coterie used can have a significant impact on system reliability. They also developed some heuristic algorithms for making vote assignments [17] and a partially enumerating algorithm [17, 5] for ND coteries. However, the total number of vote assignments, a specific class of coteries, is on the order of $O(2^{N^2})$, and the total number of ND coteries is on the order of $O(2^{2^{CN}})$, where N is the number of nodes in the system and C is a constant, so that it is not feasible to use the enumerating algorithm for a system with more than five nodes [17]. Some other heuristic vote assignment algorithms are given in [2].

To analyze the availability of the mutual exclusion mechanism, let the availability of $node_i$ be a_i. The availability depends not only on the reliability of node itself, but also on the reliability of links and other nodes. To simplify our discussion, we assume that all of the a_i's are mutually independent.

The availability of the mutual exclusion mechanism in the overall system may be calculated easily by a recursive manner if the coterie used is derived from HT. The logical hierarchy implicitly constructed in the HT transformation process reveals the reliability hierarchy. That is, for certain groups, each of those virtual nodes associated with a subhierarchy may be treated as a unit (a virtual component) that has the reliability of a submutual-exclusion mechanism. For example, let the availability of five nodes be a_1, a_2, a_3, a_4, a_5, respectively. Assume that nodes 3,4,5 become a subsystem W, and associated with the coterie $R = \{\{3,4\},\{3,5\},\{4,5\}\}$ and nodes $1, 2, W$ is a system associated with the coterie $S = \{\{1,2\},\{1,W\},\{2,W\}\}$. The overall coterie derived from the NT transformation is $S' = \mathrm{HT}(S, W, R) = \{\{1,2\},\{1,3,4\},\{1,3,5\},\{1,4,5\},\{2,3,4\},\{2,3,5\},\{2,4,5\}\}$. The availability of the mutual exclusion mechanism of S' can be defined by a function known as Triple Module Redundancy (TMR), and is given by

$$\mathrm{TMR}(S') = a_1{}^* a_2{}^* a_W + a_1{}^* a_2{}^* (1 - a_W) + a_1{}^* (1 - a_2){}^* a_W$$

$$+ (1 - a_1){}^* a_2{}^* a_W$$

where $a_W = \mathrm{TMR}(R) = a_3{}^* a_4{}^* a_5 + a_3{}^* a_4{}^* (1 - a_5) + a_3{}^* (1 - a_4){}^* a_5 + (1 - a_3){}^* a_4{}^* a_5$.

The R and S are isomorphic since, by proper renaming, they may be shown to be equivalent. The idea behind this is two-out-of-three or TMR (Triple Module Redundancy). Assuming that only the TMR coteries are adopted in HT for both original and primitive coteries, hierarchical TMR structures will be constructed. We found that only the components with availability > 0.5 can enhance the availability of a system with the technique of hierarchical TMR. Furthermore, only the components with a similar availability should be grouped into TMR to enhance the availability. That is, sometimes TMR will fail to increase the reliability. For example, grouping three components with availability 0.900, 0.700, 0.600, respectively, into a TMR will lead to 0.834, which is inferior to 0.900, the one-component system.

Consider the solution space complexity of the hierarchical TMR problem given N (odd) nonhomogeneous nodes. The problem can be divided into two phases: one to determine the structure, and the other to determine node assignment. Let $No = (N - 1)/2$, i.e., the number of virtual

nodes in the result structure; the number of nonisomorphic structures is

$$S(No) = \sum_{A=0}^{\left\lfloor \frac{No-1}{3} \right\rfloor} \sum_{B=A}^{\left\lfloor \frac{No-1-A}{2} \right\rfloor} S(A)^{*}S(B)^{*}S(No-1-A-B),$$

where $S(0) = S(1) = S(2) = 1$. The equation is derived by the following facts. No matter what structure there is for N real nodes, it will always result in $No = (N-1)/2$ virtual nodes. Solving the problem of how many structures existed for organizing N nodes is equivalent to showing how many structures existed for organizing No virtual nodes among a ternary tree hierarchy. The $S(No)$ is the summation of productions of all of the nonisomorphic three (subproblem) partitions summing up $No-1$. (Some examples are shown in the Appendix.) Even assuming that the node assignment can be solved efficiently, the $S(No)$ alone increases exponentially. Therefore, there exists no possibility to search the solution space exhaustively for a large heterogeneous system.

A simple heuristic algorithm is proposed to construct hierarchical TMR aimed at maximizing reliability. The basic idea is drawn from the Huffman tree for code compression. Hierarchical TMR will only compose a system with an odd number of components. When a system with an even number of components is given, the least reliable component will be discarded. Providing N, an odd number, components, the algorithm first sorts these components into the decreasing sequence by their reliability measure. Then it tries to put the least reliable three components into TMR by testing whether the TMR will outperform the highest reliable component among the three. If not, the algorithm just discards the last two components, and proceeds to the same algorithm to the $N-2$ components that are left. If the TMR is more reliable than the highest reliable components among the three, a new virtual component standing for the TMR will be inserted into the component sequence by the availability of TMR, and the three components will be discarded from the sequence.

Algorithm 3 (construct hierarchical TMR)

Construct-H-TMR(N)

Step 1: Sort the N components into a list with the sequence of decreasing reliability.

Step 2: If N = even then, discard the last component, $N = N-1$.

Step 3: If Reliability(TMR(a_{N-1}, a_{N-2}, a_{N-3})) $\leqslant$ Max(a_{N-1}, a_{N-2}, a_{N-3}),

then discard components $(N-1),(N-2)$,
$N=N-2$;

else group components $(N-1),(N-2),(N-3)$
into new TMR,
insert new virtual component (new TMR)
into list by its reliability,
discard the last three components in new
TMR, $N=N-2$.

Step 4: If $N \geqslant 3$, goto Step 3.

Step 5: End.

As an example, given an availability list List = {0.64, 0.67, 0.63, 0.62, 0.68, 0.58}, the algorithm sorts the List and discards the node with 0.58 reliability. The List becomes {0.68, 0.67, 0.64, 0.63, 0.62}. Then the reliability of TMR(0.64, 0.63, 0.62), 0.690632 compares favorably with 0.64, the maximum among the three. The List becomes {0.690632 = TMR(0.64, 0.63, 0.62), 0.68, 0.67} which represents a three-node system. The TMR of these three nodes is more reliable than 0.690632, so the List becomes {0.758649 = TMR(0.690632 = TMR(0.64, 0.63, 0.62, 0.68, 0.67)}.

Algorithm 3 given above determines both the TMR structure and the node assignment in one. It gives good hierarchical TMR structures and coterie. However, a question of interest arises: Is the derived coterie optimal? (The Huffman tree constructed by a similar algorithm is optimal.) We found that if the constructed TMR structure is a straight level structure, i.e., two real components in each level, except for the last level with three components, it is optimal. (Proof: interchanging two nodes belonging to different levels will lead to lower availability.) Otherwise, the resultant coterie is likely to be, but may not, optimal. For example, there are nine nodes with different availability:

Case 1

0.84, 0.80, 0.78, 0.76, 0.73, 0.72, 0.70, 0.56, 0.54.

Algorithm 3 will discard the last two nodes since TMR(0.70, 0.56, 0.54) = 0.6490. It then comes into a TMR(0.73, 0.72, 0.70) = 0.8048. A virtual node is inserted between 0.84 and 0.80. The algorithm now groups the last three nodes into TMR(0.80, 0.78, 0.76) = 0.8763. Finally, TMR(0.8763, 0.84, 0.8048) = 0.9326. We now give a solution that achieves higher availabil-

ity than this one. That is, {0.9329 = TMR(0.8446 = TMR(0.8, 0.73, 0.72), 0.8409 = TMR(0.78, 0.76, 0.7), 0.84)}. The corresponding TMR structure is shown in Figure 5(a). This coterie is equivalent to the one defined by the binary tree quorum [1].

Case 2

$$0.84, 0.82, 0.78, 0.76, 0.73, 0.72, 0.70, 0.65, 0.62.$$

The solution will be a level structure like {0.9378 = TMR(0.8819 = TMR (0.8158 = TMR(0.73, 0.7278 = TMR(0.7, 0.65, 0.62), 0.72), 0.78, 0.76), 0.84,

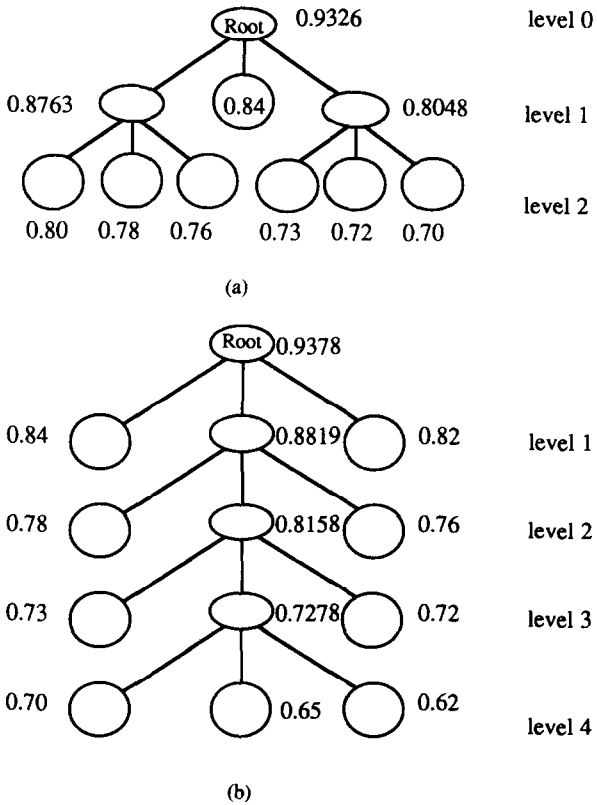


Fig. 5. Two solutions given by Algorithm 3: (a) Case 1, (b) Case 2.

0.82)}, which is illustrated in Figure 5(b). In this case, the assignment of nodes is optimal.

Providing a system in which the components (nodes) have the same availability and $N = 9$, the algorithm will result in an RH(3,3) structure, the same as HQC [5], defined in Section 4. The structures derived in these cases are special cases belonging to the regular structures defined in Section 4. Note that irregular structures will occur as well.

Given that a highly reliable component and a number of low reliable components are given, it is interesting to see whether the algorithm put the highly reliable component in the proper location in the TMR hierarchy. The proposed algorithm gives a very good answer compared to the situation that fixed structures such as HQC or tree structures are employed; the HQC will treat all nodes equally for it is a fully distributed scheme (they assumed a homogeneous system in [8]), and in tree quorum [1], they only suggest that the higher reliable node should be placed closer to the root. The result of our proposed algorithm may be interpreted as: by hierarchical TMR, a number of less reliable components can achieve the same availability as a highly reliable component and can compete for the same level in the hierarchy.

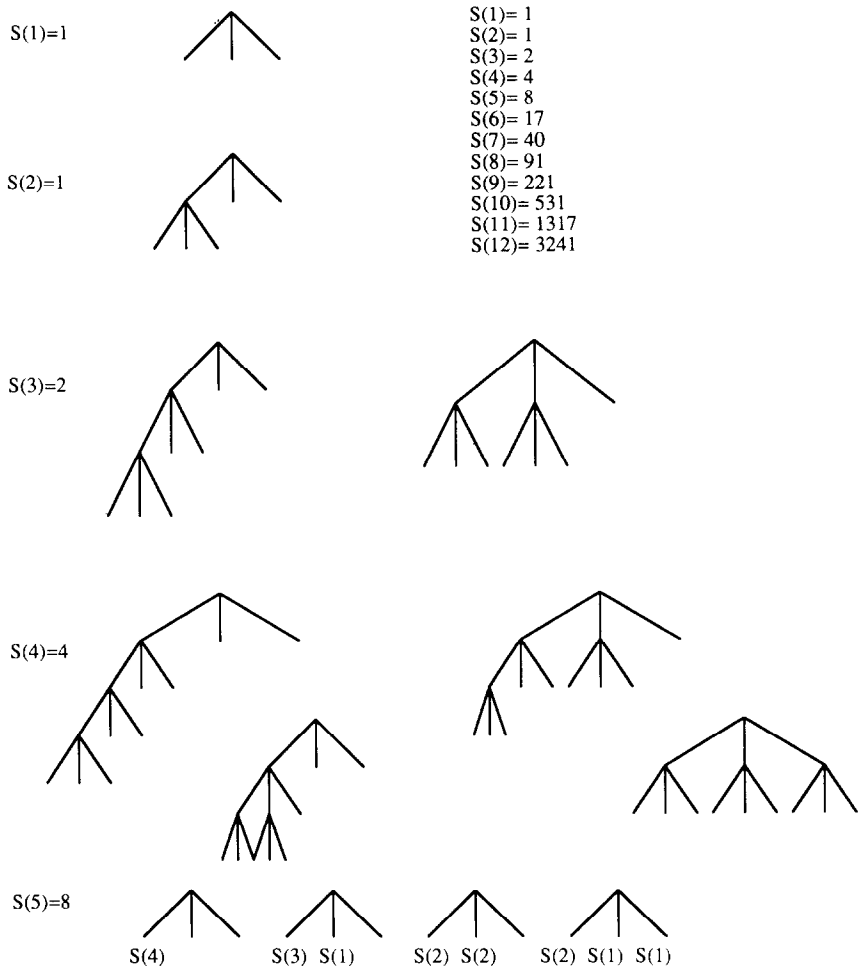
5. CONCLUSIONS

Mutual exclusion is crucial for the design of distributed systems. Many quorum schemes have been proposed to solve the problem. Coteries are general schemes for mutual exclusion solutions. Garcia-Molina and Barbara developed a transformation algorithm (TA) to transform one nondominated (ND) coterie to another. The TA may properly be used for partially enumerating ND coterie. However, the TA proposed in [5] is not appropriate for the case of large coterie design because properties of the new produced ND coterie are not easily predictable. This paper describes a hierarchical transformation exploiting hierarchy to generate scalable coteries from smaller coteries that can be used to solve the distributed mutual exclusion problem. This provides a simple way to produce new larger ND coteries through transformation from smaller ND coteries. By systematically applying HT, coteries for a large number of nodes may be obtained. Furthermore, the new generated coterie will be characterized by the original coterie, primitive coteries, and the corresponding hierarchical structure. This makes the transformation more predictable than trial and error. In the context of heterogeneous systems, a heuristic algorithm is proposed to construct a hierarchical TMR structure in which HT may be applied to generate good coterie design. With these methods, the mutual

exclusion mechanism designers may design tailored quorum schemes based on different performance measurements and the underlying system characteristics by taking advantage of the hierarchical structure and the HT transformation.

APPENDIX

THE STRUCTURE COMPLEXITY



S(1)= 1
S(2)= 1
S(3)= 2
S(4)= 4
S(5)= 8
S(6)= 17
S(7)= 40
S(8)= 91
S(9)= 221
S(10)= 531
S(11)= 1317
S(12)= 3241

REFERENCES

1. D. Agrawal and A. El Abbadi, An efficient and fault-tolerant algorithm for distributed mutual exclusion, *ACM Trans. Comput. Syst.* 9(1):1–20 (Feb. 1991).
2. D. Barbara and H. Garcia-Molina, The reliability of voting mechanisms, *IEEE Trans. Comput.* 36(10):1197–1208 (Oct. 1987).
3. D. Cheriton and W. Zwaenepoel, Distributed process groups in the V kernel, *ACM Trans. Comput. Syst.* 3(2):77–107 (1985).
4. S. Y. Cheung et al., Multi-dimensional voting: A general method for implementing synchronization in distributed systems, in *Proceedings of the Tenth International Conference on Distributed Computer Systems*, 1990, pp. 326–369.
5. H. Garcia-Molina and D. Barbara, How to assign votes in a distributed system, *J. ACM* 32(4):841–860 (Oct. 1985).
6. D. K. Gifford, Weight voting for replicated data, in *Proceedings of the Seventh ACM SIGOPS Symposium Operating System Principles*, 1979, pp. 150–159.
7. J. M. Helary, N. Plouzeau, and M. Raynal, A distributed algorithm for mutual exclusion in an arbitrary network, *J. Computer* 31(4):289–295 (1988).
8. A. Kumar, Hierarchical quorum consensus: A new algorithm for managing replicated data, *IEEE Trans. Comput.* 40(9):996–1004 (Sept. 1991).
9. L. Lamport, Time, clocks, and the ordering of events in a distributed system, *Commun. ACM* 21(7):145–159 (July 1978).
10. L. Lamport, The implementation of reliable distributed multiprocessor systems, *Comput. Networks* 2:95–114 (1978).
11. M. Maekawa, A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems, *ACM Trans. Comput. Syst.* 3(2):145–159 (May 1985).
12. J. Misra, Detecting termination of distributed computations using markers, in *Proceedings Second ACM Symposium on Principles of Distributed Computing*, 1985, pp. 237–249.
13. K. Raymond, A tree-based algorithm for distributed mutual exclusion, *ACM Trans. Comput. Syst.* 7(1):61–77 (1989).
14. G. Ricart and A. K. Agrawal, An optimal algorithm for mutual exclusion in computer networks, *Commun. ACM* 24(1):9–17 (Jan. 1981).
15. B. A. Sanders, The information structure of distributed mutual exclusion algorithms, *ACM Trans. Comput. Syst.* 5:284–299 (Aug. 1987).
16. I. Suzuki and T. Kasami, A distributed mutual exclusion algorithm, *ACM Trans. Comput. Syst.* 3(4):344–349 (Nov. 1985).
17. Z. Tong and R. Y. Kain, Vote assignments in weighted voting mechanisms, *IEEE Trans. Comput.* 40(5):664–667 (May 1991).
18. R. H. Thomas, A majority consensus approach to concurrency control, *ACM Trans. Database Syst.* 4(2):180–209 (June 1979).

Received 10 March 1992; revised 2 October 1992