Minimizing the Maximum Delay for Reaching Consensus in Quorum-Based Mutual Exclusion Schemes

Tatsuhiro Tsuchiya, *Member*, *IEEE*, Masatoshi Yamaguchi, and Tohru Kikuno, *Member*, *IEEE*

Abstract—The use of quorums is a well-known approach to achieving mutual exclusion in distributed computing systems. This approach works based on a *coterie*, a special set of node groups where any pair of the node groups shares at least one common node. Each node group in a coterie is called a *quorum*. Mutual exclusion is ensured by imposing that a node gets consensus from all nodes in at least one of the quorums before it enters a critical section. In a quorum-based mutual exclusion scheme, the delay for reaching consensus depends critically on the coterie adopted and, thus, it is important to find a coterie with small delay. In [5], Fu introduced two related measures called *max-delay* and *mean-delay*. The former measure represents the largest delay among all nodes, while the latter is the arithmetic mean of the delays. She proposed polynomial-time algorithms for finding max-delay and mean-delay optimal coteries when the network topology is a tree or a ring. In this paper, we first propose a polynomial-time algorithm for finding max-delay optimal coteries and, then, modify the algorithm so as to reduce the mean-delay of generated coteries. Unlike the previous algorithms, the proposed algorithms can be applied to systems with arbitrary topology.

Index Terms—Quorums, coteries, communication delay, mutual exclusion, distributed systems.

1 Introduction

The distributed mutual exclusion problem is to guarantee that at most one computing node can enter a critical section at a time. This problem is widely recognized as one of the most fundamental problems in distributed computing since it arises in various kinds of distributed systems. For example, consider a system in which its nodes share an exclusive resource. If two or more nodes try to access the resource, a conflict occurs. To avoid such a situation, mutual exclusion has to be ensured.

The use of *coteries* is known as an elegant approach to mutual exclusion in distributed systems. A coterie is a special set of node groups such that any two node groups have at least one node in common (*intersection property*) and no node group is a superset of any other node group (*minimality property*) [4]. Node groups in a coterie are called *quorums*. Given a coterie, mutual exclusion can be achieved as follows: Before entering the critical section, a node has to acquire permission from all the nodes in at least one quorum. On the other hand, each node is allowed to give its permission to at most one node. By the intersection property, it is guaranteed that no more than one node can enter the critical section at a time.

Since the performance or robustness of such a mutual exclusion scheme depends critically on the coterie adopted by the scheme, many researchers have studied methods of designing coteries that optimize various objective functions [7], [8], [9], [14], [15]. In addition to availability and message complexity [12], the communication delay needed for achieving quorum consensus is also recognized as an important factor. Especially for systems requiring short response time, such as replicated database systems, minimizing the delay for reaching consensus is a very significant task.

Recently, the notions of *max-delay* and *mean-delay* of coteries have been introduced by Fu [5]. The max-delay of a coterie is the maximum of the delays among all nodes, while the mean-delay is the average. Fu has shown that there must be a delay-optimal coterie in a special subset of coteries, called *nondominated (ND) coteries*. Based on this result, she has proposed polynomial-time algorithms to find max-delay optimal and mean-delay optimal coteries for systems with special topologies, such as trees and rings. Since the number of ND coteries in such networks is very small, the algorithms can efficiently find delay-optimal coteries by enumerating ND coteries.

However, finding delay-optimal coteries on general graphs has been left as an open problem. The difficulty of this problem is mainly due to the fact that enumerative approaches are impractical for systems with general topology. This is because the number of ND coteries explodes when the number of nodes exceeds only five [4]. Recently, Bioch and Ibaraki developed an enumeration method based on Boolean algebra [1]. According to [1], however, the maximum size of the system that the method was able to handle was only seven nodes.

In this paper, we consider the problem of finding max-delay optimal coteries in systems with arbitrary

- *T. Tsuchiya and T. Kikuno are with the Department of Informatics and Mathematical Science, Osaka University, 1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan.*
  E-mail: {t-tutiya, kikuno}@ics.es.osaka-u.ac.jp.
- *M. Yamaguchi is with Sumitomo Corp.*

Manuscript received 26 May 1998.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 106899.

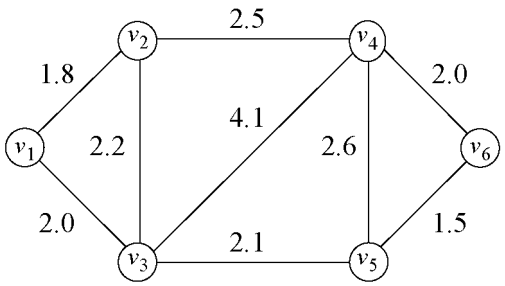

Fig. 1. An example of a system.

topology. We propose two algorithms to solve this problem. These algorithms take a completely different approach from Fu's in the sense that they do not use the notion of ND coteries. The first algorithm we present finds a max-delay optimal coterie on an arbitrary network with time complexity $O(n^3 \log n)$, where n is the number of nodes. The second algorithm is a modification of the first algorithm. By incorporating heuristics, this algorithm finds a max-delay optimal coterie with smaller mean-delay than that found by the first algorithm. The time complexity of the second algorithm is also $O(n^3 \log n)$.

The remainder of the paper is organized as follows: In Section 2, we explain the system model and basic notions about coteries. In Section 3, we propose an algorithm to find max-delay optimal coteries. In Section 4, we extend the proposed algorithm in order to generate max-delay optimal coteries with smaller mean-delay. In Section 5, we show the results obtained by applying the proposed algorithms to several sample systems. Finally, we conclude this paper with a brief summary in Section 6.

2 Preliminaries

We adopt the system model used in [5]. We consider a distributed system modeled by an edge-weighted undirected graph $G = (V, E, w)$, where $V = \{v_1, v_2, \cdots, v_n\}$ is a set of n nodes and E is a set of edges. We assume that G is connected and has neither self-loop nor parallel edges. Each edge $e \in E$ is assigned a positive real number $w(e)(> 0)$ as the weight of the edge. Fig. 1 illustrates an example of a system. The number attached to each edge represents its weight.

For two distinct nodes v_i and v_j, the virtual distance $dist(v_i, v_j)(= dist(v_j, v_i))$ between the two nodes is defined as the length of the shortest path on G, where the length of a path is the sum of the weights assigned to the edges in the path. The virtual distance represents the communication delay between the two nodes. In Fig. 1, for example, $dist(v_1, v_6)$ is 5.6. We assume that, for any node v_i, the virtual distance $dist(v_i, v_i)$ is zero.

Given a subset V' of V, we assume that a node v_i can communicate with all members of V' with delay $\max_{v \in V'}\{dist(v_i, v)\}$. We let $delay(v_i, V')$ denote this delay, i.e.,

$$delay(v_i, V') = \max_{v \in V'}\{dist(v_i, v)\},$$

where V' is a nonempty subset of V.

In the following, we give the formal definition of a coterie. The notion of coterie was first introduced by Garcia-Molina and Barbara [4].

Definition 1 (Coterie). *A coterie $\mathcal{C}$ is a set of nonempty subsets of V such that the following conditions hold:*

1. *(Intersection property) $Q \cap Q' \neq \emptyset$ for any pair $Q, Q' \in \mathcal{C}$.*
2. *(Minimality property) There is no pair $Q, Q' \in \mathcal{C}$ such that $Q \subset Q'$.*

Each element in a coterie is called a quorum. Given a coterie, mutual exclusion can be achieved using some distributed mutual exclusion algorithms [2], [9], [13]. In the following, we use the term *node group* to refer to a nonempty subset of V.

Example 1. Let the set of all nodes V be $\{v_1, v_2, v_3, v_4, v_5, v_6\}$. Now, consider the following sets of node groups, $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$, and $\mathcal{C}_4$.

$$\mathcal{C}_1 = \{\{v_1\}\}$$
$$\mathcal{C}_2 = \{\{v_2, v_4\}, \{v_2, v_5\}, \{v_4, v_5\}\}$$
$$\mathcal{C}_3 = \{\{v_1, v_2, v_3\}, \{v_4, v_5, v_6\}\}$$
$$\mathcal{C}_4 = \{\{v_1\}, \{v_1, v_2, v_3\}\}.$$

Among the four sets of node groups, $\mathcal{C}_1$ and $\mathcal{C}_2$ are coteries. $\mathcal{C}_3$ is not a coterie because it does not satisfy the intersection property. $\mathcal{C}_4$ is not a coterie either because it does not satisfy the minimality property.

Adopting definitions from [5], we now introduce the notions of the delay of a node in a coterie and the max-delay and the mean-delay of a coterie. The delay of node v_i in coterie $\mathcal{C}$, or $Delay(v_i, \mathcal{C})$, is given by

$$\begin{aligned} Delay(v_i, \mathcal{C}) &= \min_{Q \in \mathcal{C}}\{\max_{v \in Q}\{dist(v_i, v)\}\} \\ &= \min_{Q \in \mathcal{C}}\{delay(v_i, Q)\}, \end{aligned}$$

and the max-delay and the mean-delay of coterie $\mathcal{C}$ are given by

$$max{-}delay(\mathcal{C}) = \max_{v \in V}\{Delay(v, \mathcal{C})\}$$
$$mean{-}delay(\mathcal{C}) = \frac{1}{|V|}\sum_{v \in V} Delay(v, \mathcal{C}).$$

Intuitively, the delay of a node in a coterie is the communication delay for achieving consensus when the node accesses its nearest quorum. The max-delay of a coterie is the maximum of the delays among all nodes and the mean-delay of a coterie is the arithmetic mean of the delays of all nodes.

Example 2. Consider the system shown in Fig. 1. Let $\mathcal{C} = \{\{v_2, v_4\}(= Q_1), \{v_2, v_5\}(= Q_2), \{v_4, v_5\}(= Q_3)\}$ be a coterie in the system. Now, take the node v_1 as an example. Among the three quorums, Q_2 is the nearest quorum from v_1, and $Delay(v_1, \mathcal{C})$ is

```
Begin
/* Step 1. Calculate the virtual distance between every pair of nodes */
  Execute an algorithm such as [6] or [10].

/* Step 2. Find min by binary search */
  Sort {dist(v_i, v_j) | v_i, v_j ∈ V} in ascending order and store in a_1, a_2, ..., a_m
  il := 1, iu := m
  While (il ≠ iu) do
    Begin
      i := ⌊(il + iu)/2⌋
      If Intersection_Check(a_i) = true then iu := i else il := i + 1
    End
  D_i := {v ∈ V | dist(v_i, v) ≤ a_il} for all v_i ∈ V /* min = a_il, D_i = NB_i(min) */

/* Step 3. Remove all supersets from NB_i(min)'s. */
  tmp := a list of D_i's
  For all pairs v_i, v_j ∈ V do
    Begin
      If D_i ⊆ D_j then remove D_j from tmp
      If D_j ⊂ D_i then remove D_i from tmp
    End
  C_opt := a set of node groups in tmp /* max-delay optimal coterie */
End

Function Intersection_Check(r)
Begin
  D_i := {v ∈ V | dist(v_i, v) ≤ r} for all v_i ∈ V
  For all pairs v_i, v_j ∈ V do
    If D_i ∩ D_j = ∅ then return(false)
  return(true)
End
```

Fig. 2. The proposed algorithm for obtaining a max-delay optimal coterie.

$$delay(v_1, Q_2) = dist(v_1, v_5) = 4.1.$$

For v_2, v_3, v_4, v_5, and v_6, the nearest quorums are Q_1, Q_2, Q_1, Q_3, and Q_3, respectively. Since

$$\begin{aligned} Delay(v_1, \mathcal{C}) &= 4.1, \\ Delay(v_2, \mathcal{C}) &= 2.5, \\ Delay(v_3, \mathcal{C}) &= 2.2, \\ Delay(v_4, \mathcal{C}) &= 2.5, \\ Delay(v_5, \mathcal{C}) &= 2.6, \end{aligned}$$

and

$$Delay(v_6, \mathcal{C}) = 2.0,$$

$max{-}delay(\mathcal{C})$ and $mean{-}delay(\mathcal{C})$ are 4.1 and 2.65, respectively.

Now, a max-delay optimal coterie is defined as follows:

Definition 2 (Max-delay optimal coterie). *Let $\mathcal{UC}$ be the set of all coteries. A coterie $\mathcal{C} \in \mathcal{UC}$ is said to be max-delay optimal if and only if*

$$max{-}delay(\mathcal{C}) = \min_{\mathcal{C}' \in \mathcal{UC}} \{max{-}delay(\mathcal{C}')\}.$$

3 Finding Max-Delay Optimal Coteries

As mentioned before, the problem we consider in this paper is to find a max-delay optimal coterie for a given $G = (V, E, w)$.

Suppose that virtual distance $dist(v_i, v_j)$ for any $v_i, v_j \in V$ has been computed from G. In fact, this is possible in polynomial time by using some previously proposed algorithms (e.g., [10]). We can then restate the problem by explicitly specifying a quorum from which each node gets consensus.

Problem 1. Given V and virtual distance $dist(v_i, v_j)$ for any $v_i, v_j \in V$, find an n-tuple of node groups $(N_1, N_2, \cdots, N_n)$ that minimizes $\max_{v_i \in V}\{delay(v_i, N_i)\}$, under the conditions that 1) for any two groups N_i and N_j, $N_i \cap N_j \neq \emptyset$, and 2) for any two groups N_i and N_j, $N_i \not\subset N_j$.

Suppose $(N_1, N_2, \cdots, N_n)$ is an optimal solution to Problem 1 and let $\mathcal{C}$ be the set of all N_is $(i = 1, 2, \cdots, n)$. Note that the number of elements in $\mathcal{C}$ is not necessarily n since N_i may be equal to N_j for some $i, j(\neq i)$. Then, $\mathcal{C}$ is obviously a max-delay optimal coterie.

In order to solve this problem, we consider a more tractable problem, removing Condition 2) from Problem 1.

Problem 2. Given V and virtual distance $dist(v_i, v_j)$ for any $v_i, v_j \in V$, find an n-tuple of node groups $(N_1, N_2, \cdots, N_n)$ that minimizes $\max_{v_i \in V}\{delay(v_i, N_i)\}$, under the condition that 1) for any two node groups N_i and N_j, $N_i \cap N_j \neq \emptyset$.

Lemma 1. *Suppose that $(N_1, N_2, \cdots, N_n)$ is an optimal solution to Problem 2. Let $\mathcal{C}$ be the set of N_is such that no other $N_j (i \neq j)$ is a proper subset of N_i, i.e.,*

$$\mathcal{C} = \{N_i | N_i \in \mathcal{N} \text{ such that } \forall N_j \in \mathcal{N} - \{N_i\}, N_j \not\subset N_i\},$$

where $\mathcal{N}$ is the set of all N_is $(i = 1, 2, \cdots, n)$. (Note that the number of elements in $\mathcal{N}$ is not necessarily n since N_i may be equal to N_j for some $i, j(\neq i)$.) Then, $\mathcal{C}$ is a max-delay optimal coterie.

Proof. By definition, it is clear that $\mathcal{C}$ is a coterie. If N_i is not in $\mathcal{C}$, then there is another node group N_j in $\mathcal{C}$ such that $N_j \subset N_i$. Now, consider another n-tuple $(M_1, M_2, \cdots, M_n)$ such that if $N_i \in \mathcal{C}$, then $M_i = N_i$; otherwise, M_i is equal to a node group N_j in $\mathcal{C}$ such that $N_j \subset N_i$. Then, since $M_i \subseteq N_i$ holds for any i, $delay(v_i, M_i) \leq delay(v_i, N_i)$ also holds for any i. It is then clear that

$$\max_{v_i \in V}\{delay(v_i, M_i)\} \leq \max_{v_i \in V}\{delay(v_i, N_i)\}.$$

In addition, since $(N_1, N_2, \cdots, N_n)$ is an optimal solution to Problem 2,

$$\max_{v_i \in V}\{delay(v_i, M_i)\} \geq \max_{v_i \in V}\{delay(v_i, N_i)\}.$$

Hence, $\max_{v_i \in V}\{delay(v_i, M_i)\} = \max_{v_i \in V}\{delay(v_i, N_i)\}$. This implies that $(M_1, M_2, \cdots, M_n)$ is also an optimal solution to Problem 2.

Since Problem 2 differs from Problem 1 only in that it does not impose Condition 2) on solutions, the inequality $\max_{v_i \in V}\{delay(v_i, M_i)\} \leq \max_{v_i \in V}\{delay(v_i, L_i)\}$ holds if $(L_1, L_2, \cdots, L_n)$ is an optimal solution to Problem 1. Notice that $(M_1, M_2, \cdots, M_n)$ satisfies Conditions 1) and 2) in Problem 1. Hence, it is also an optimal solution to Problem 1. By definition, $\mathcal{C}$ is equal to the set of M_is, thus, $\mathcal{C}$ is a max-delay optimal coterie. □

Now, we define $NB_i(r)$ as the set of all nodes whose virtual distance from v_i is equal to or smaller than r, i.e., $NB_i(r) = \{v \in V | dist(v_i, v) \leq r\}$. In addition, let min be the smallest positive real number such that for any $v_i, v_j \in V$, $NB_i(min)$ and $NB_j(min)$ intersect, i.e., $NB_i(min) \cap NB_j(min) \neq \emptyset$. Then, the following lemma holds.

Lemma 2. *$(NB_1(min), NB_2(min), \cdots, NB_n(min))$ is an optimal solution to Problem 2.*

Proof. (By contradiction) If

$$(NB_1(min), NB_2(min), \cdots, NB_n(min))$$

is not an optimal solution to Problem 2, there must exist another n-tuple $(N_1, N_2, \cdots, N_n)$ such that $N_i \cap N_j \neq \emptyset$ for any i, j and $min > \max_{v_i \in V}\{delay(v_i, N_i)\}$. Let

$$d = \max_{v_i \in V}\{delay(v_i, N_i)\}.$$

Then, since $N_i \subseteq NB_i(d)$ holds for any i, $NB_i(d) \cap NB_j(d) \neq \emptyset$ for any i, j. Because

$$\max_{v_i \in V}\{delay(v_i, NB_i(d))\} = d,$$

$(NB_1(d), NB_2(d), \cdots, NB_n(d))$ is also an optimal solution to Problem 2. Since $d < min$, this is a contradiction to the definition of min. □

Let $\mathcal{C}_{opt}$ be the set of $NB_i(min)$s such that no $NB_j(min)(i \neq j)$ is a proper subset of $NB_i(min)$, i.e.,

$$\mathcal{C}_{opt} = \{NB_i(min) | NB_i(min) \in \mathcal{NB} \text{ such that } \forall NB_j(min) \in \mathcal{NB} - \{NB_i(min)\}, NB_j(min) \not\subset NB_i(min)\},$$

where $\mathcal{NB}$ is the set of all $NB_i(min)$s $(i = 1, 2, \cdots, n)$. Then, we obtain the following theorem.

Theorem 1. *$\mathcal{C}_{opt}$ is a max-delay optimal coterie.*

Proof. The proof is clear from Lemma 1 and Lemma 2. □

Theorem 1 leads to an algorithm for finding a max-delay optimal coterie. Fig. 2 shows this algorithm. It consists of three steps.

In Step 1, the virtual distance between every pair of nodes is calculated. This can be done, for example, by using Floyd's classical algorithm with time complexity $O(n^3)$ [6]. (There are also faster algorithms, such as [10].)

In Step 2, min and $NB_i(min)$s are calculated. At the beginning of this step, elements of $\{dist(v_i, v_j) | v_i, v_j \in V\}$,

```
/* Step 2'. Reduce the sizes of D_i's */
  POOL := {(D_i, v_j) | 1 ≤ i ≤ n, v_j ∈ D_i}
  While (POOL ≠ ∅) do
    Begin
      /* Select one tuple (D_i, v_j) */
      From POOL, choose (D_i, v_j) such that
        dist(v_i, v_j) = max_{(D_i,v_j)∈POOL} {dist(v_i, v_j)}.
        (If their is more than one such tuple, choose a tuple
        with the largest value of |D_i|.)
      POOL := POOL − (D_i, v_j)
      /* Check whether (D_i, v_j) can be removed or not */
      If (∀D_{k(≠i)}, D_k ∩ (D_i − {v_j}) ≠ ∅) then
        D_i := D_i − {v_j}
    End
```

Fig. 3. Step 2' of the modified algorithm.

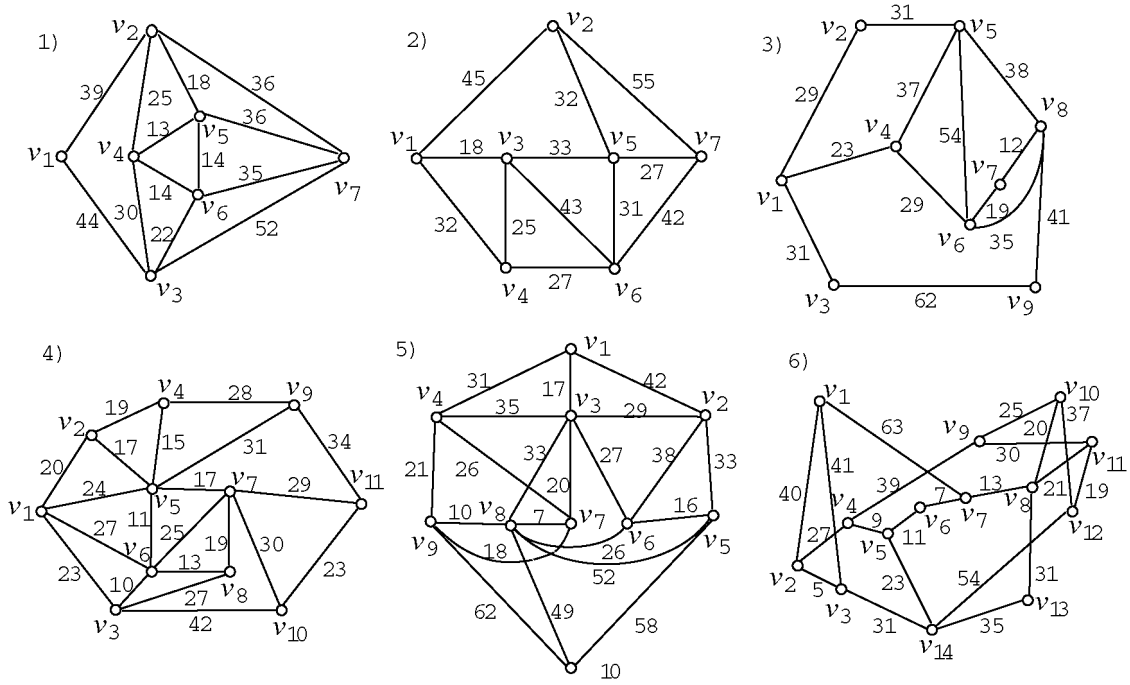

Fig. 4. Sample systems.

TABLE 1
Max-Delay Optimal Coteries

System	Original Algorithm	Modified Algorithm
1	$\{\{v_1, v_2, v_3\}, \{v_2, v_5, v_6, v_7\}, \{v_1, v_3, v_4, v_5, v_6\}, \{v_2, v_3, v_4, v_5, v_6\}\}$	$\{\{v_2, v_3\}, \{v_2, v_4\}, \{v_2, v_6\}, \{v_3, v_4, v_6\}\}$
2	$\{\{v_1, v_2, v_5\}, \{v_5, v_6, v_7\}, \{v_1, v_3, v_4, v_6\}\}$	$\{\{v_1, v_5\}, \{v_1, v_6\} \{v_5, v_6\}\}$
3	$\{\{v_3, v_7, v_8, v_9\}, \{v_1, v_2, v_3, v_4, v_5\}, \{v_1, v_2, v_3, v_4, v_9\}, \{v_1, v_4, v_5, v_6, v_7, v_8\}, \{v_4, v_5, v_6, v_7, v_8, v_9\}\}$	$\{\{v_3, v_4\}, \{v_1, v_3, v_5\}, \{v_1, v_4, v_7\}, \{v_3, v_7, v_8\}, \{v_4, v_5, v_7\}, \{v_4, v_5, v_8\}\}$
4	$\{\{v_7, v_{10}, v_{11}\}, \{v_4, v_5, v_9, v_{11}\}, \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}\}$	$\{\{v_5, v_7\}, \{v_5, v_{11}\}, \{v_7, v_{11}\}\}$
5	$\{\{v_7, v_8, v_{10}\}, \{v_1, v_3, v_4, v_7, v_8, v_9\}, \{v_1, v_2, v_3, v_5, v_6, v_7, v_8\}, \{v_2, v_3, v_5, v_6, v_7, v_8, v_9\}\}$	$\{\{v_2, v_8\}, \{v_3, v_8\}, \{v_7, v_8\}, \{v_2, v_3, v_7\}\}$
6	$\{\{v_1, v_2, v_3, v_7\}, \{v_5, v_6, v_7, v_8, v_{10}, v_{11}, v_{13}, v_{14}\}, \{v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{14}\}, \{v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}\}, \{v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_{12}, v_{13}, v_{14}\}, \{v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{14}\}\}$	$\{\{v_2, v_7\}, \{v_4, v_7\}, \{v_6, v_7\}, \{v_7, v_{14}\}, \{v_2, v_4, v_6, v_{14}\}\}$

that is, all candidates for min, are sorted in ascending order and stored in $a_1, a_2, \cdots$. Using this data structure and Function Intersection_Check, min can be obtained by a binary search. Function Intersection_Check checks whether $NB_i(r) \cap NB_j(r) \neq \emptyset$ holds for any i, j or not.

TABLE 2
Max-Delay and Mean-Delay Values of the Obtained Coteries

System	Max-delay	Mean-delay	
		Original algorithm	Modified algorithm
1	44.0	36.86	28.57
2	51.0	40.86	37.86
3	62.0	58.33	50.56
4	41.0	34.09	28.91
5	56.0	45.90	30.50
6	63.0	55.79	36.71

Since the number of candidates for min is at most $\frac{n(n-1)}{2}$, in the worst case, it takes $O(n^2 \log n^2)$ time to sort them by merge sort or heap sort. The while loop in Step 2 is iterated $O(\log n^2)$ times. Function Intersection_Check requires $O(n^3)$ time per invocation because the for loop in this function is repeated at most $\frac{n(n-1)}{2}$ times and the if statement checking whether or not $D_i \cap D_j$ is empty takes $O(n)$ time. Hence, it takes $O(n^3 \log n)$ time to complete the while loop. Thus, the time complexity of Step 2 is $O(n^3 \log n)$.

In Step 3, all supersets are removed from $NB_i(min)$s. The remaining elements form a max-delay optimal coterie. The time complexity of this step is $O(n^3)$ because each if statement takes $O(n)$ time and the for loop is iterated $\frac{n(n-1)}{2}$ times. Consequently, it is seen that the time complexity of this algorithm is $O(n^3 \log n)$.

Example 3. Consider the system shown in Fig. 1. The proposed algorithm works as follows: In Step 1, the virtual distance between every pair of nodes is calculated. Then, in Step 2, the values of min and $NB_i(min)$s are calculated. First, candidates for min are sorted as follows:

$$1.5,\ 1.8,\ 2.0,\ 2.1,\ 2.2,\ 2.5,\ 2.6,\ 3.6,\ 4.1,\ 4.3,\ 4.5,\ 5.6.$$

Using a binary search, the value of min is calculated. In this case, $min = 3.6$. Then, each $NB_i(min)$ is calculated and stored in D_i as follows:

$$\begin{aligned} D_1 &= \{v_1, v_2, v_3\} \\ D_2 &= \{v_1, v_2, v_3, v_4\} \\ D_3 &= \{v_1, v_2, v_3, v_5, v_6\} \\ D_4 &= \{v_2, v_4, v_5, v_6\} \\ D_5 &= \{v_3, v_4, v_5, v_6\} \\ D_6 &= \{v_3, v_4, v_5, v_6\}. \end{aligned}$$

In Step 3, we obtain a max-delay optimal coterie by removing all supersets from the $NB_i(min)$s. In this case, the max-delay optimal coterie is

$$\{\{v_1, v_2, v_3\}, \{v_2, v_4, v_5, v_6\}, \{v_3, v_4, v_5, v_6\}\}.$$

Obviously, the max-delay of the coterie is 3.6, while its mean-delay is 2.533.

4 Extension for Reducing Mean-Delay

The algorithm presented in the previous section can successfully find a max-delay optimal coterie $\mathcal{C}_{opt}$ for any given system G. However, there may be other max-delay optimal coteries whose mean-delay is smaller than $\mathcal{C}_{opt}$. In order to find such a max-delay optimal coterie with smaller mean-delay, we propose a new algorithm by modifying the original algorithm.

Specifically, we insert a new step (referred to as Step 2′) into the original algorithm between Step 2 and Step 3. At the end of Step 2, $NB_1(min)$, $NB_2(min)$, $\cdots$, $NB_n(min)$ have been obtained. Each $NB_i(min)$ is stored in D_i in the algorithm shown in Fig. 2. In Step 2′, we examine the nodes in $D_1, D_2, \cdots, D_n$ one by one, in a certain order, and remove those nodes whose removal does not destroy the property that every pair of D_i and D_j intersect. Even with such an additional stage that reduces the sizes of the D_is, it is guaranteed that the coterie obtained is max-delay optimal and its mean-delay is not larger than $\mathcal{C}_{opt}$. The following theorem ensures this.

Theorem 2. *Let $(N_1, N_2, \cdots, N_n)$ be an n-tuple of node groups such that $N_i \cap N_j \neq \emptyset$ for any $v_i, v_j \in V$ and $N_i \subseteq NB_i(min)$ for any $v_i \in V$. Let $\mathcal{C}$ be the set of N_is such that no other $N_j (i \neq j)$ is a proper subset of N_i, i.e.,*

$$\mathcal{C} = \{N_i | N_i \in \mathcal{N} \text{ such that } \forall N_j \in \mathcal{N} - \{N_i\}, N_j \not\subset N_i\},$$

where $\mathcal{N}$ is the set of all N_is $(i = 1, 2, \cdots, n)$. Then, $\mathcal{C}$ is a max-delay optimal coterie and mean-delay$(\mathcal{C}) \leq$ mean-delay$(\mathcal{C}_{opt})$.

Proof. Since $N_i \subseteq NB_i(min)$ for any i, $delay(v_i, N_i) \leq delay(v_i, NB_i(min))$ holds for any i. By Lemma 2, $(NB_1(min), NB_2(min), \cdots, NB_n(min))$ is an optimal solution to Problem 2. Hence, $\max_{v_i \in V}\{delay(v_i, N_i)\} = min$ and $(N_1, N_2, \cdots, N_n)$ is also an optimal solution to Problem 2. By Lemma 1, it is then clear that $\mathcal{C}$ is a max-delay optimal coterie. By definition, for any quorum $Q \in \mathcal{C}_{opt}$, there is a $Q' \in \mathcal{C}$ such that $Q' \subseteq Q$. Note that if $Q' \subseteq Q$, then, for any node v_i, $delay(v_i, Q') \leq delay(v_i, Q)$. Hence, $mean\text{-}delay(\mathcal{C}) \leq mean\text{-}delay(\mathcal{C}_{opt})$. □

```
/* Step 2'. Reduce the sizes of the D_i's */
  /* Initialize Variables */
  For all v_i ∈ V do I_i := {j|v_i ∈ D_j} /* D_j's that include v_j */
  For all 1 ≤ i, j ≤ n do ns_{i,j} := |D_i ∩ D_j|
    /* number of nodes shared by D_i and D_j */
  For all v_i ∈ V do
    Begin
      Sort all v_j ∈ D_i in descending order of dist(v_i, v_j)
      and store them in a list pool_i
      rest_i := |D_i| /* number of elements in D_i */
    End
  /* Reduce the sizes of the D_i's */
  While (∃i, pool_i ≠ ∅) do
    Begin
      /* Select one tuple (D_i, v_j) */
      max := −1
      For all 1 ≤ k ≤ n such that pool_k ≠ ∅ do
        Begin
          v_l := the first node in pool_k
          If (dist(v_k, v_l) > max) or
            ((dist(v_k, v_l) = max) and (rest_k > rest_i))
          Then begin
            i := k, j := l
          End
        End
      Remove v_j from the top of pool_i
      /* Check whether v_j can be removed from D_i or not */
      flag := true
      For all D_k such that k ∈ I_j and k ≠ i
        If ns_{i,k} ≤ 1 then
          Begin
            flag := false, Break the for loop
          End
      /* Remove v_j from D_i */
      If (flag = true) then
        Begin
          D_i := D_i − {v_j}, I_j := I_j − {i}, rest_i := rest_i − 1
          For all D_k such that k ∈ I_j do
            Begin
              ns_{i,k} := ns_{i,k} − 1, ns_{k,i} := ns_{k,i} − 1
            End
        End
  End
```

Fig. 5. The details of Step 2' of the modified algorithm.

In order to determine which nodes are checked and removed, we do the following: Let the tuple (D_i, v_j) denote a node v_j in D_i. (There are then $\sum_{v_i \in V} |D_i|$ tuples that have to be considered.) We choose (D_i, v_j) with the largest value of $dist(v_i, v_j)$ first, since this is most intuitive. If there is more than one tuple with the largest value, a node is chosen from D_i such that $|D_i|$ is the largest. Fig. 3 shows Step 2' of the modified algorithm. (Steps 1, 2, and 3 are omitted since

they are exactly the same as the original algorithm shown in Fig. 2.)

Since the total number of tuples considered is less than n^2, the while loop in Step 2′ is repeated no more than n^2 times. It takes $O(n)$ time to select one tuple during each iteration of the while loop by sorting the nodes in each D_i according to their distances from v_i. Unlike in the first algorithm, whether or not the intersection property continues to hold after removal of the selected tuple can be determined in $O(n)$ time by maintaining the number of nodes shared by every pair of D_is. (The details of Step 2′ are shown in the Appendix.) Hence, the time complexity of this step is $O(n^3)$. Since the original algorithm (the one without Step 2′) is of $O(n^3 \log n)$ time complexity, the modified algorithm is also of $O(n^3 \log n)$ time complexity.

Example 4. Consider the system shown in Fig. 1. Since Steps 1 and 2 of the modified algorithm are the same as the original algorithm, each D_i in the modified algorithm has been set to $NB_i(min)$ at the end of Step 2, as shown in the previous example. Thus, there are 24 tuples that have to be considered in this case.

Step 2′ reduces the sizes of the D_is as follows: First, among the 24 tuples, tuple (D_i, v_j) is selected such that the value of $dist(v_i, v_j)$ is the largest. In this case, (D_3, v_6) and (D_6, v_3) have the greatest value, $3.6 (= dist(v_3, v_6) = dist(v_6, v_3))$. Since $|D_3| (= 5)$ is larger than $|D_6| (= 4)$, (D_3, v_6) is chosen first. Because $D_3 - \{v_6\}$, i.e., $\{v_1, v_2, v_3, v_5\}$ has at least one common node with each of D_1, D_2, D_4, D_5, and D_6, v_6 is removed from D_3. Then, the D_is become as follows:

$$
\begin{aligned}
D_1 &= \{v_1, v_2, v_3\} \\
D_2 &= \{v_1, v_2, v_3, v_4\} \\
D_3 &= \{v_1, v_2, v_3, v_5\} \\
D_4 &= \{v_2, v_4, v_5, v_6\} \\
D_5 &= \{v_3, v_4, v_5, v_6\} \\
D_6 &= \{v_3, v_4, v_5, v_6\}.
\end{aligned}
$$

Next, (D_6, v_3) is chosen for checking. If v_3 were removed from D_6, D_6 would no longer intersect with D_1. Thus, this node is not removed. (D_4, v_5) is chosen next since $dist(v_4, v_5)$ is the largest among the remaining unchecked tuples. This process is repeated until all tuples have been checked. Consequently, the D_is become as follows:

$$
\begin{aligned}
D_1 &= \{v_2, v_3\} \\
D_2 &= \{v_2, v_3\} \\
D_3 &= \{v_2, v_3\} \\
D_4 &= \{v_2, v_6\} \\
D_5 &= \{v_3, v_6\} \\
D_6 &= \{v_3, v_6\}.
\end{aligned}
$$

A max-delay optimal coterie is derived from the D_is in Step 3. In this case, the coterie is $\{\{v_2, v_3\}, \{v_2, v_6\}, \{v_3, v_6\}\}$. The mean-delay of this coterie is 2.433. (Recall that the mean-delay of the coterie obtained by the original algorithm was 2.533.)

5 Evaluation

Using the C language, we coded the original algorithm and the modified algorithm. For Step 1, we adopted Floyd's algorithm [6]. We took a collection of networks from [3], as shown in Fig. 4, and used them to represent the topologies of sample systems. Random weights were assigned to the edges. For each system in Fig. 4, we executed the programs on a SUN Ultra 1 workstation and obtained max-delay optimal coteries. In all cases, the running time needed for generating a max-delay optimal coterie was less than 0.1 second.

Table 1 shows the max-delay optimal coteries obtained by the two algorithms. Table 2 summarizes the max-delays and mean-delays of the coteries. These results clearly show that max-delay optimal coteries generated by the modified algorithm have much smaller mean-delays than those generated by the original algorithms. For System 6, for example, the mean-delay of the coterie generated by the modified algorithm is about 35 percent smaller than that of the coterie generated by the original algorithm.

6 Conclusions

In this paper, we have proposed two algorithms to find max-delay optimal coteries for systems with arbitrary topology. The first algorithm finds a max-delay optimal coterie on an arbitrary network with $O(n^3 \log n)$ time complexity, where n is the number of nodes. The second algorithm is a modification of the first algorithm. By incorporating heuristics, this algorithm finds a max-delay optimal coterie with smaller mean-delay than the original algorithm. The time complexity of the second algorithm is also $O(n^3 \log n)$. Through case studies, we have shown that this modification can lower the mean-delay effectively.

The modified algorithm is an approximation method for solving the problem of finding a max-delay optimal coterie such that its mean-delay is minimized. Unfortunately, we have not yet developed an optimal algorithm to solve this problem nor do we know the complexity of this problem. Further, how to find mean-delay optimal coteries in arbitrary networks is still left as an open problem. In addition, evaluation of the proposed algorithms using measures other than delay, such as availability or load [11], also needs further study.

Appendix

Fig. 5 shows the details of Step 2′ of the modified algorithm.

Acknowledgments

The authors wish to thank Professor Ichiro Suzuki of the University of Wisconsin-Milwaukee for his useful comments. We also express our gratitude to the anonymous reviewers for useful comments, especially on the complexity of the proposed algorithms. This work is in part supported by a grant from Inamori Foundation.

REFERENCES

[1] J.C. Bioch and T. Ibaraki, "Generating and Approximating Nondominated Coteries," *IEEE Trans. Parallel and Distributed Systems,* vol. 6, no. 9, pp. 905-914, Sept. 1995.

[2] G. Cao, M. Singhal, Y. Deng, N. Rishe, and W. Sun, "A Delay-Optimal Quorum-Based Mutual Exclusion Scheme with Fault-Tolerance Capability," *Proc. 18th Int'l Conf. Distributed Computing Systems,* pp. 444-451, 1998.

[3] Y.G. Chen and M.C. Yuang, "A Cut-Based Method for Terminal-Pair Reliability," *IEEE Trans. Reliability,* vol. 45, no. 3, pp. 413-416, 1996.

[4] H. Garcia-Molina and D. Barbara, "How to Assign Votes in a Distributed System," *J. ACM,* vol. 32, no. 4, pp. 841-860, Oct. 1985.

[5] A.W. Fu, "Delay-Optimal Quorum Consensus for Distributed Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 8, no. 1, pp. 59-69, Jan. 1997.

[6] R.W. Floyd, "Algorithm 97: Shortest Path," *Comm. ACM,* vol. 5, p. 345, 1962.

[7] T. Harada and M. Yamashita, "Nondominated Coteries on Graphs," *IEEE Trans. Parallel and Distributed Systems,* vol. 8, no. 6, pp. 667-672, June 1997.

[8] T. Ibaraki, H. Nagamochi, and T. Kameda, "Optimal Coteries for Rings and Related Networks," *Distributed Computing,* vol. 8, pp. 191-201, 1995.

[9] M. Maekawa, "A $\sqrt{N}$ Algorithm for Mutual Exclusion in Decentralized Systems," *ACM Trans. Computer Systems,* vol. 3, no. 2, pp. 145-159, May 1985.

[10] A. Moffat and T. Takaoka, "An All Pairs Shortest Path Algorithm with Expected Running Time $O(n^2 \log n)$," *SIAM J. Computing,* vol. 16, no. 6, pp. 1,023-1,031, Dec. 1987.

[11] M. Naor and A. Wool, "The Load, Capacity, and Availability of Quorum Systems," *SIAM J. Computing,* vol. 27, no. 2, pp. 423-447, Apr. 1998.

[12] D. Saha, S. Rangarajan, and S.K. Tripathi, "An Analysis of the Average Message Overhead in Replica Control Protocols," *IEEE Trans. Parallel and Distributed Systems,* vol. 7, no. 10, pp. 1,026-1,034, Oct. 1996.

[13] B. Sanders, "The Information Structure of Distributed Mutual Exclusion Algorithm," *ACM Trans. Computer Systems,* vol. 5, no. 3, pp. 284-299, Aug. 1987.

[14] M. Spasojevic and P. Berman, "Voting as the Optimal Static Pessimistic Scheme for Managing Replicated Data," *IEEE Trans. Parallel and Distributed Systems,* vol. 5, no. 1, pp. 64-73, Jan. 1994.

[15] J. Tang and N. Natarajan, "Obtaining Coteries That Optimize the Availability of Replicated Databases," *IEEE Trans. Knowledge and Data Eng.,* vol. 5, no. 2, pp. 309-321, Apr. 1993.

Tatsuhiro Tsuchiya received ME and PhD degrees in computer engineering from Osaka University in 1995 and 1998, respectively. He is currently a research associate in the Department of Informatics and Mathematical Science at Osaka University. His research interests are in the areas of distributed computing and fault-tolerant computing. He is a member of the IEEE.

Masatoshi Yamaguchi received a BE degree in computer engineering from Osaka University in 1998. Currently, he is working for Sumitomo Corporation.

Tohru Kikuno received MS and PhD degrees from Osaka University in 1972 and 1975, respectively. He was with Hiroshima University from 1975 to 1987. Since 1990, he has been a professor in the Department of Informatics and Mathematical Science at Osaka University. His research interests include the quantitative evaluation of software development processes, the analysis and design of fault-tolerant systems, and the design of procedures for testing communication protocols. He served as a program co-chair of the First International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC '98) and the Fifth International Conference on Real-Time Computing Systems and Applications (RTCSA '98).