On Coterie for Generalized Distributed Resource Allocation Algorithm *

Shao-Chin Sung
School of Information Science, JAIST
son@jaist.ac.jp

Yoshifumi Manabe
NTT Basic Research Laboratories
manabe@theory.brl.ntt.co.jp

Abstract

This paper discusses the generalized resource allocation problem defined by H. Kakugawa and M. Yamashita. A set of processes shares a set of resources of an identical type. Each process may have different accessible resources. Each resource must be used by at most one process at any time. They proposed a coterie-based distributed algorithm for this problem, however their algorithm does not guarantee the requirement that the resource allocation for the set of processes with no common accessible resources must be performed without any interference. In order to guarantee the requirement, this paper defines a new structure, *sharing structure coterie*. We show a necessary and sufficient condition of the existence of a sharing structure coterie. The decision of the existence of a sharing structure coterie with respect to a given distributed system is NP-complete. We also show a distributed resource allocation algorithm which guarantees the above requirement for distributed systems whose sharing structure coteries do not exist or are difficult to obtain.

1 Introduction

In many *distributed systems*, processes shares some common resources, such as files, memory, and printers. Multiple processes must not access the same resource at the same time. A *resource allocation problem* is to guarantee that each resource is accessed by at most one process at any time. Many distributed algorithms for this problem, which include coterie-based algorithms [3] have been presented for the case when there is one unit of shared resource.

Fujita et al. [2], Manabe et al. [9], and Baldoni [1] considered k-mutual exclusion problem in which there are k (≥ 1) identical shared resources and all of the resources are accessible to every process. They defined k-coterie for the k-mutual exclusion problem.

Kakugawa and Yamashita [5] defined a generalization of the above problem, which is called the *generalized resource allocation problem* in this paper. Each process may have different accessible resources. They proposed a distributed algorithm for the problem using a new class of coteries, called *local coteries*. Since there may exist a set of processes with no common accessible resource, it is a natural requirement that the resource allocation for the set of processes must be performed without any interference. However, their algorithm does not guarantee this requirement.

In order to guarantee the above requirement, this paper introduces a new class of coteries, which is called *sharing structure coteries*. By using a sharing structure coterie instead of a local coterie, the distributed resource allocation algorithm proposed in [5] guarantees the above requirement.

We show a necessary and sufficient condition of existence of a sharing structure coterie. The decision of the existence of a sharing structure coterie for a given distributed system is NP-complete. We also show a distributed resource allocation algorithm which guarantees the above requirement for distributed systems whose sharing structure coteries do not exist or are difficult to obtain.

2 Local coteries

The model of a *distributed system* and the *generalized resource allocation problem* are defined in the same way as in [5]. A *distributed system* consists of a set of processes U and a set of resources R which are shared by processes in U. Any two processes are connected by a bidirectional communication link. Information exchange between the processes is based on message-passing through the link. The delivery of messages may have unpredictable finite delay, but the order of messages is unchanged. Both the processes and the links are assumed to be error-free. Each process has its own local clock.

A resource accessible to process u is called a *acces-*

*Part of this work was done while the first author was staying at NTT Basic Research Laboratories.

Information Systems and Technologies for Network Society, Fukuoka, Japan, September 1997

sible resource of process u. The set of all accessible resources of u is denoted by $\alpha(u) \subseteq R$ for each $u \in U$. Triple $S = (U, R, \alpha)$ is called the *sharing structure* of the system. A *generalized resource allocation problem* for a given sharing structure $S = (U, R, \alpha)$ is the problem to allocate the resources according to requests. The allocation must satisfy the following conditions.

Allocation validity: The resources which are accessed by process u are in $\alpha(u)$ for any process $u \in U$.

Mutual exclusion: Each resource $r \in R$ is accessed by at most one process at the same time.

Coterie is introduced in [3] for a distributed resource allocation algorithm (i.e., $|R| = 1$ and $\alpha(u) = R$ for all $u \in U$). *Coterie* Q is a family of subset of U, i.e., $Q \subseteq 2^U$, that satisfies the following properties:

Non-emptiness: $\forall q \in Q\ [q \neq \emptyset]$.
Minimality: $\forall q, r \in Q\ [q \not\subseteq r]$.
Intersection property: $\forall q, r \in Q\ [q \cap r \neq \emptyset]$.

An element of a coterie is called a *quorum*. The coterie-based algorithm for the resource allocation problem [8, 10] can simply described as follows: Determine a coterie Q. Initially, each process has one "permission". If a process u wants to access the resource, it arbitrarily selects a quorum $q \in Q$, and sends a request to each process in q. Then, u waits to receive a permission from each process in q, and it accesses the resource. After the access, u releases the resource and returns the permission to each process in q. The intersection property guarantees the mutual exclusion condition.

To make the algorithm deadlock-free and starvation-free, a priority of requests is introduced. The request with the smallest timestamp has the highest priority. Suppose a process u had sent its permission to a process v's request whose timestamp is T_v, however v has not received enough permissions. If u receives a request from a process w which has timestamp $T_w < T_v$, then v has to return the permission to u, and u sends the permission to w.

In the generalized resource allocation problem, each process has a different set of accessible resources. Thus Kakugawa and Yamashita introduced a new coterie, which is called a *local coteries*. A local coterie $\{Q_u \subseteq 2^U \mid u \in U\}$ for sharing structure S satisfies the following properties:

Non-emptiness: $\forall u \in U\ [Q_u \neq \emptyset]$.
Minimality: $\forall u \in U, \forall q, r \in Q_u\ [q \not\subseteq r]$.
Intersection property: $\forall u, v \in U$,
$[\alpha(u) \cap \alpha(v) \neq \emptyset \Rightarrow \forall q \in Q_u, \forall r \in Q_v\ [q \cap r \neq \emptyset]]$.

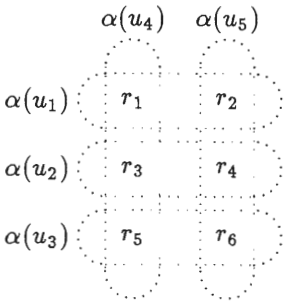

Figure 1: Sharing structure S.

They showed a construction of a local coterie for any given sharing structure S.

Example: Consider a sharing structure $S = (U, R, \alpha)$, where $U = \{u_1, \ldots, u_5\}$, $R = \{r_1, \ldots, r_6\}$, and

$\alpha(u_i) = \{r_{2i-1}, r_{2i}\}$ for $i = 1, 2, 3$,
$\alpha(u_4) = \{r_1, r_3, r_5\}$, and
$\alpha(u_5) = \{r_2, r_4, r_6\}$ (see Figure. 1).

The local coterie $\{Q_u \mid u \in U\}$ for S is as follows.

$Q_{u_i} = \{\{u_i, u_4, u_5\}\}$ for $i = 1, 2, 3$, and
$Q_{u_i} = \{\{u_i, u_1, u_2, u_3\}\}$ for $i = 4, 5$.

The coterie-based distributed algorithm for the generalized resource allocation problem that uses a local coterie in [5] is similar to the one for the resource allocation problem. Instead of coterie, process u uses local coterie Q_u. Instead of a permission, a state list is sent to the requesting process, where the state list is a list of current states of all resources. Process u has the right to lock and access to a set of resources $R_u \subseteq \alpha(u)$ if the following conditions are satisfied:

- Process u receives a state list from each process in an arbitrary quorum of Q_u.
- States of all resources in R_u are free in every state list.

The correctness of the resource allocation algorithm is proved. However, unnecessary waiting among processes might occur. In the above example, consider the case that only u_1 and u_2 want to access the resources at the same time. Since $\alpha(u_1) \cap \alpha(u_2) = \emptyset$, these two requests do not block each other. However, $q_{u_1} \cap q_{u_2} = \{u_4, u_5\} \neq \emptyset$. Thus, u_4 and u_5 first send the state list to the higher priority request, say, u_1. After the state list update by u_1, the updated state list is sent to u_2 and u_2 can use the resources. When there exist k resources, there can be such a process waiting chain whose length is $O(k)$. This waiting is unnecessary since the processes have no common accessible resource.

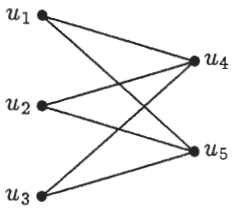

Figure 2: Sharing structure graph G_S.

3 Sharing Structure Coteries

In order to guarantee no such unnecessary waiting in the resource allocation, we define a *sharing structure coterie* which satisfies the three properties of a local coterie (i.e., non-emptiness, minimality, and intersection property) and the following property:

Disjointness property: $\forall u, v \in U$,

$$[\alpha(u) \cap \alpha(v) = \emptyset \Rightarrow \forall q \in Q_u,\ \forall r \in Q_v\ [q \cap r = \emptyset]].$$

By using a sharing structure coterie, the algorithm proposed in [5] guarantees that the resource allocation for the set of processes with no common accessible resource must be performed without any interference because of the disjointness property.

Unfortunately, there exist some sharing structures that have no sharing structure coteries. The necessary and sufficient condition of the existence of a sharing structure coterie for a given sharing structure is shown in the following.

Definition 1 *For a given sharing structure $S = (U, R, \alpha)$, sharing structure graph $G_S = (U, E_S)$ is an undirected graph, where*

$$E_S = \{(u, v) \in U \times U \mid u \neq v,\ \alpha(u) \cap \alpha(v) \neq \emptyset\}.$$

A sharing structure graph for the sharing structure of the example in the previous section is shown in Fig. 2.

The existence condition is related to an NP-complete problem, "COVERING BY CLIQUE" [4] for the sharing structure graph G_S.

Definition 2 *A clique cover of a graph $G = (V, E)$ is a collection of subsets $V_1, \ldots, V_k$ of V such that,*

- *each V_i induces a complete subgraph of G, and*
- *for each edge $(u, v) \in E$, there exists some V_i that contains both u and v.*

Given a graph G and a positive integer $K \leq |E|$, the "COVERING BY CLIQUE" problem is:

"Is there a clique cover of G with cardinality $k \leq K$? "

Theorem 3 *For any sharing structure S, there exists a sharing structure coterie with respect to S if and only if there exists a clique cover of the sharing structure graph G_S with cardinality at most $|U|$.* ∎

Proof. The keyword conflict problem [6] is as follows: Given an $n \times n$ zero-one matrix $P = [p_{ij}]$ such that $p_{ij} = p_{ji}$ and $p_{ii} = 1$ for all i and j, obtain an $n \times m$ zero-one matrix $Z = [z_{ij}]$ such that for all s and t $(z_{s1}z_{t1}, z_{s2}z_{t2}, \ldots, z_{sm}z_{tm}) = (0, 0, \ldots 0)$ if and only if $p_{st} = 0$.

Matrix P can be constructed from a sharing structure S as follows. Let N be an arbitrary injective mapping from $\{1, 2, \ldots, |U|\}$ to U. $p_{ij} = 1$ if and only if $\alpha(N(i)) \cap \alpha(N(j)) \neq \emptyset$. It is obvious that $p_{ij} = p_{ji}$ and $p_{ii} = 1$ for all i and j.

Matrix Z can be constructed from a sharing structure coterie $\mathcal{Q} = \{Q_u \mid u \in U\}$ with respect to S as follows. $z_{ij} = 1$ if and only if $N(j) \in q$ for some $q \in Q_{N(i)}$. From the intersection property and disjointness property, for all s and t $(z_{s1}z_{t1}, z_{s2}z_{t2}, \ldots, z_{sm}z_{tm}) = (0, 0, \ldots 0)$ if and only if $p_{st} = 0$.

Thus, obtaining a sharing structure coterie is equivalent to the keyword conflict problem with the restriction $m \leq |U|$. The keyword conflict problem with the restriction $m \leq K$ for a given K has been proved to be NP-complete since it is equivalent to obtaining a clique cover on G_S whose cardinality is at most K [7]. Therefore, obtaining a sharing structure coterie with respect to S is also equivalent to obtaining a clique cover of the sharing structure graph G_S with cardinality at most $|U|$ and it is NP-complete. ∎

For the sharing structure S in the example, there is no sharing structure coterie with respect to S, since the sharing structure graph G_S has a unique clique cover with with cardinality $6 > |U| = 5$ (see Fig. 2).

Now we show a construction of a sharing structure coterie from a clique cover $\mathcal{C}_S = \{c_1, \ldots, c_m\} \subseteq 2^U$ of a sharing structure graph G_S with cardinality $m \leq |U|$.

Let $p_u = \{c \in \mathcal{C}_S \mid u \in c\}$ for each $u \in U$. Let σ be an arbitrary injective mapping from $\mathcal{C}_S$ to U. Note that such an injective mapping σ exists, since $|\mathcal{C}_S| = m \leq |U|$. Then, a sharing structure coterie $\mathcal{Q} = \{Q_u \mid u \in U\}$ for sharing structure S is defined as $Q_u = \{q_u\}$, where

$$q_u = \{\sigma(c) \mid c \in p_u\}\ .$$

Note that if u is an isolated node, $Q_u = \{\{\}\}$.

It can be easily shown that this coterie satisfies the properties of a sharing structure coterie.

4 Distributed Resource Allocation Algorithm

As shown in Theorem 3, there is no sharing structure coterie for some sharing structures. Even if there is such a coterie, it is sometimes very difficult to obtain since the decision of the existence of a sharing structure coterie is NP-complete. Then, we consider a distributed resource allocation algorithm for sharing structures whose sharing structure coterie do not exist or are difficult to obtain.

For any sharing structure $S = (U, R, \alpha)$, suppose a clique cover $\mathcal{C}$ for the sharing structure graph G_S is given, where $|\mathcal{C}| > |U|$. Note that it is not implying that no sharing structure coterie with respect to S exists. Consider a sharing structure $S' = (U \cup W, R, \alpha')$, where $U \cap W = \emptyset$ and $|U \cup W| = |\mathcal{C}|$,

- $\alpha'(u) = \alpha(u)$ for all $u \in U$, and
- $\alpha'(w) = \emptyset$ for all $w \in W$.

Then, $\mathcal{C}$ is also a clique cover for sharing structure graph $G_{S'}$ of S', since each $w \in W$ is an isolated node in $G_{S'}$. From Theorem 3, there exists a sharing structure coterie with respect to S'. Thus, for a given sharing structure S and a given clique cover $\mathcal{C}$ of the sharing structure graph G_S, the resource allocation for S can be performed by the algorithm in [5] using a sharing structure coterie for S', where the processes in W are simulated by the processes in U.

The analysis of the message complexity for this algorithm is the same as the one in [5] (see [5] for detail). In the best case, the message complexity for one request is $4|q|$, where q is the smallest quorum. In the worst case, the message complexity for one request is $(7 + |\alpha(u)|)|q'|$, where q' is the largest quorum.

By the local coterie construction algorithm in [5], the size of process u's quorum, $|q_u|$, satisfies $|q_u| = |\{v \in U \mid \alpha(u) \cap \alpha(v) \neq \phi\}|$.

By the sharing structure coterie construction algorithm, the size of u's quorum q'_u, $|q'_u|$, satisfies $|q'_u| \leq |\{v \in U \mid v \neq u, \alpha(u) \cap \alpha(v) \neq \phi\}| < |q_u|$. Though $u \in q_u$ and u does not have to send a message to itself, the number of messages sent using a sharing structure coterie is not larger than that using a local coterie.

Since the size of every quorum is not larger than the one in [5], the message complexity of this algorithm is not worse than the one in [5].

5 Conclusion

We have defined a new class of coteries, sharing structure coteries, for the distributed resource allocation algorithm which guarantees that resource allocations for processes with no common accessible resource are performed without any interference.

We showed the existence condition of a sharing structure coterie with respect to any sharing structure. It implies that the decision of the existence of a sharing structure coterie is NP-complete.

For sharing structures whose sharing structure coterie do not exist or are difficult to obtain, we showed that the distributed resource allocation can be performed by the algorithm in [5] using a sharing structure coterie for a sharing structure with imaginary processes.

Acknowledgments. The authors would like to thank Prof. Masafumi Yamashita of Hiroshima University for pointing out the paper of Ref. [7]. They also thank Dr. Hirofumi Katsuno of NTT Basic Research Laboratories for his encouragement and suggestions.

References

[1] R. Baldoni. An $O(N^{M/(M+1)})$ distributed algorithm for the k-out of-m resource allocation problem. In *Proc. 14th IEEE Int. Conf. on Distributed Computing Systems*, pages 81–87, 1994.

[2] S. Fujita, M. Yamashita, and T. Ae. Distributed k-mutual exclusion problem and k-coteries. In *Proc. 2nd Int. Symposium on Algorithms*, pages 22–31, 1991. LNCS 557.

[3] H. Garcia-Molina and D. Barbara. How to assign votes in a distributed system. *Journal of the ACM*, 32(4):841–860, 1985.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability: a guide of the theory of NP-completeness.* W. H. Freeman and Company, San Francisco, 1979.

[5] Hirotsugu Kakugawa and Masafumi Yamashita. Local coteries and a distributed resource allocation algorithm. *Transactions of Information Processing Society of Japan*, 37(8):1487–1498, 1996.

[6] E. Kellerman. Determination of keyword conflict. *IBM Tech. Disclosure Bull.*, 16(2):544–546, 1973.

[7] L.T. Kou, L.J. Stockmeyer, and C.K. Wong. Covering edges by cliques with regard to keyword conflicts and intersection graphs. *Communication of ACM*, 21(2):135–139, 1978.

[8] M. Maekawa. A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems. *ACM Transactions on Computer Systems*, 3(2):145–159, 1985.

[9] Yoshifumi Manabe and Shigemi Aoyagi. A distributed k-mutual exclusion algorithm using k-coterie. Technical Report COMP 93-43, IEICE, 1993.

[10] M. Singhal and N. G. Shivaratri. *Advanced Concepts in Operating Systems- Distributed, Database, and Multiprocessor Operating Systems.* McGraw-Hill, 1994.