k-Arbiter: A Safe and General Scheme for h-out of-k Mutual Exclusion Problems

Roberto Baldoni, Yoshifumi Manabe, Michel Raynal, Shigemi Aoyagi

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

k-Arbiter: A safe and general Scheme for h-out of-k mutual exclusion problems

Roberto Baldoni, Yoshifumi Manabe
Michel Raynal, Shigemi Aoyagi

N° 2523
April 1995

PROGRAMME 1

ISSN 0249-6399

k-Arbiter: A safe and general Scheme for h-out of-k mutual exclusion problems *

Roberto Baldoni**, Yoshifumi Manabe ***
Michel Raynal**, Shigemi Aoyagi***

Programme 1 — Architectures parallèles, bases de données, réseaux et systèmes distribués
Projets Adp

Rapport de recherche n°2523 — April 1995 — 18 pages

Abstract: Mutual exclusion is a well-known problem that arise when multiple processes compete, in an uncoordinated way, for the acquisition of shared resources over a distributed system. In particular, k-mutual exclusion allows at most k processes to get one unit of the same resource simultaneously. These paradigms do not cover all the cases in which resource accesses must be serialized over a distributed system. There exist cases (e.g. the bandwidth of communication lines) where the amount of shared resource might differ from request to request (for example, audio and video communications). In this paper, we formalize this problem as the *h-out of-k mutual exclusion problem*, in which each request concerns some number h ($1 \leq h \leq k$) of units of shared resource and no unit is allocated to multiple processes at the same time. We present a general scheme for a *quorum-based h-out of-k mutual exclusion algorithm* that relies on a collection of quorums called *k-arbiter*. Several examples of k-arbiters are discussed, two particular classes of k-arbiters are investigated and a metric to evaluate the resiliency with respect to failures of k-arbiters is also given.

Key-words: Mutual exclusion, distributed synchronization, quorums, coteries, k-coteries.

(Résumé : tsvp)

*Work partially supported by the following Basic Research Action Programs of the European Community: the HCM project under contract No.3702 "CABERNET" and the ESPRIT project under conder contract No. 6360 "BROADCAST".

**IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France. Roberto Baldoni is on leaving from Dipartimento di Informatica e Sistemistica, University of Rome "La Sapienza", via Salaria 113, I-00198 Rome, Italy. This author is also supported by a grant of the *Consiglio Nazionale delle Ricerche* under contract No.93.02294.CT12

***NTT Basic Research Laboratories, 3-9-11 Midori-cho,Musashino-shi, Tokio 180, Japan

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

Le concept de k-arbitre: un mécanisme général pour l'exclusion mutuelle du type "h parmi k"

Résumé : Cet article étudie l'exclusion mutuelle généralisée du type *h parmi k*. Il existe k ressources identiques et chaque processus peut en demander, à tout instant, un nombre quelconque; de plus, chaque exemplaire de la ressource ne peut être utilisé que par un processus à la fois.
Une solution simple à ce problème fondée sur un nouveau concept (*k-arbitre*) est proposée. Un k-arbitre généralise la notion de coterie habituellement utilisée pour résoudre l'exclusion mutuelle simple. Des propriétés générales des k-arbitres sont établies et plusieurs familles de k-arbitres sont proposées et évaluées.

Mots-clé : Exclusion mutuelle, synchronisation répartie, quorums, coteries.

1 Introduction

The mutual exclusion problem arises when multiple processes use shared resources such as files, printers, and communication lines. Simple mutual exclusion states that only one process at any time be in its critical section which is the part of the source code in which the process executes private operations with the shared resource. In recent years, many distributed mutual exclusion algorithms have been presented [23, 20] and generalizations to k-mutual exclusion have been also considered [9, 16, 18]. The k-mutual exclusion allows at most k processes to enter the critical section at the same time. This problem corresponds to the case when there is a pool of k units of the shared resource and each process can requests at most one unit. All these mutual exclusion algorithms have been classified in two main categories [23, 20]: the quorum (or permission)-based and the token-based. In the latter a process allocates the shared resource as soon as it receives a token [8]. In the former a process gets the resource as soon as it receives an explicit permission, by means of a message, from all processes constituting its quorum; after using the resource it returns the permission to the processes of its quorum. In order to achieve the formal correctness (i.e., safety property), quorums must verify some property of mutual intersection [14], hence the concept of *coterie* [5] for simple mutual exclusion and k-coterie [10, 17, 16] for k-mutual exclusion have been used. A coterie is a set of mutually intersecting sets of processes. Although token-based algorithms generally show good performance about the number of messages exchanged to get the shared resource, they suffer from poor failure resiliency. On the contrary, quorum-based algorithms gracefully tolerate failures of nodes and network partitions [5, 6, 7, 17].

However, simple and k-mutual exclusion do not cover all possible cases in which resource accesses must be strictly serialized; for example, some applications, such as the ones using communication bandwidth, the amount of shared resource in each request might differ from request to request. For example, they would be different for a video and an audio communication request. Thus, mutual exclusion algorithms should handle requests for multiple units of the shared resource. Following [19], we formalize this problem as the *h-out of-k mutual exclusion problem*, in which each request concerns at most h ($1 \leq h \leq k$) units of the shared resource and no unit is allocated to multiple processes at the same time (i.e., safety property).

In this paper we define a general scheme for quorum-based h-out of-k mutual exclusion algorithms preserving safety. At this end, we introduce a new structure consisting of a collection of quorums, called *k-arbiter*, that subsumes the concept of coterie. Two particularly interesting classes of coteries are extended to k-arbiters and thoroughly investigated: *symmetric* and *non-dominated* k-arbiters. The first ones have the nice property to split the effort to provide quorum availability equally among all the processes and to pay the same price for each request in terms of messages exchanged [15]. The second provides high protection against failures and network partitions [5]. Moreover, several examples of k-arbiters are analyzed. The study of actions needed to ensure the liveness property (no starvation and no deadlock phenomena) in quorum-based algorithms are out of the aim of this work (readers are referred to [2, 3, 21, 16] for a thorough discussion about such actions).

The paper is structured into 3 main sections. In Section 2, the h-out of-k problem is defined and concepts of non-dominated and symmetric coteries are recalled. Section 3 shows an outline of a quorum-based distributed h-out of-k algorithm and the correctness condition that must be verified by quorums in order that the algorithm be safe. The k-arbiter concept is then introduced in Section 4 and the classes of symmetric and non-dominated k-arbiters are also defined. In the same Section, we analyze several examples of k-arbiters showing if they belong to some particular class of k-arbiters. Finally, a metric to evaluate the resiliency with respect to failures of k-arbiters is also given.

2 Basic Definitions

2.1 Distributed System

A distributed system is a set U of n processes $\{p_1, p_2, \ldots, p_n\}$ that communicate by exchanging messages. We assume that each pair of processes is connected by a logical channel, and the message delay is unpredictable but finite. Moreover, each channel is assumed to have infinite capacity, to be error-free and FIFO (messages are received in the order they were sent). Processes do not share either a common clock or a shared memory, no bound exists to the relative speed of processes and they fail according to the fail-stop model [22] and failures can be detected by other nodes.

2.2 The h-out of-k Mutual Exclusion Problem

The h-out of-k problem can be described as follows. There are k units of a resource that can be shared by processes in the following manner: at any given time each unit may be used by at most one process, and each process may have allocated some units. A process requests h ($1 \leq h \leq k$) units of the resource all at once and, to avoid deadlock phenomena, the process is blocked until it gets all of its requested units. After that, the process starts using the units and then releases them all at once. A *conflict* arises whenever one or more processes try to allocate a number of units greater than those currently available. It is clear that if $h = 1$ the h-out of-k problem corresponds to the k-mutual exclusion one, moreover if $k = 1$ we get the simple mutual exclusion.

In order to maintain the integrity of the system, h-out of-k mutual exclusion algorithm must satisfy the following properties:

safety : each unit of the resource may be used by at most one process at any given time;

liveness : all the requests are eventually satisfied.

Safety is obtained by ensuring that the following inequality always holds (where h_j is the number of units of the resource currently allocated to process p_j):

$$\sum_{j \in U} h_j \leq k, \tag{1}$$

This is the invariant associated with the h-out of-k mutual exclusion problem. A key point to obtaining safety is to detect all conflicts that may arise in the acquisition of resources. Liveness requires that the system always progresses towards the execution of a critical section (no deadlock) and that conflicts are not always resolved "against" some subset of processes (no starvation).

2.3 Quorums and Coteries

Concepts introduced in this Section were introduced [14, 5] to solve the mutual exclusion problem:

Definition 2.1 *A* quorum Q *under* U *is a non-empty subset of* U. ■

The simpler use of quorums to obtain mutual exclusion is the majority quorum algorithm [24] where a process has allocated the shared resource only when it receives a permission from the majority of the processes. Hence, the mutual exclusion property is straighforward guaranteed (only one process at a time can get the majority of permissions). The introduction of the concept of coterie allowed the refinement of the previous mutual exclusion technique [5]:

Definition 2.2 *A set of quorums* $C = \{Q_1, Q_2, \ldots, Q_m\}$ *is a* coterie under U *iff the following properties hold:*

- *(Intersection Property) For any pair of quorums* $Q_i, Q_j \in C :: Q_i \bigcap Q_j \neq \phi$;
- *(Minimality Property) For any pair of distinct quorums* $Q_i, Q_j \in C :: Q_i \not\subseteq Q_j$. ■

A process selects a quorum from a coterie and waits for receiving permissions to allocate the shared resource from all processes in the quorum. Mutual exclusion is guaranteed from the Intersection Property of quorums members of the coterie. We briefly recall some examples of coteries that will be used in the following:

1. Singleton Coteries: It is formed by single set of one element ($C = \{\{p_i\}\}$).
2. The Majority Coteries: It is formed by all the subsets of U with size $\lfloor n/2 \rfloor + 1$ where n is the number of processes.
3. Maekawa's Coteries [15]: each one of the n processes corresponds to one of the $(h-1)^2 + (h-1) + 1$ points of a *finite projective plane* whose order is $h-1$; each quorum of a coterie is associated with a line of the plane which contains h points. It is known that there exists a finite projective plane only if $h-1$ is a positive power of a prime.
4. Square Grid Coteries [15]: in this case, processes are structured into a square grid $(1 \ldots \sqrt{n}, 1 \ldots \sqrt{n})$; the quorum associated with the point (i, j) of the square grid consists of all processes located in line i and column j of the grid.

2.3.1 Non-dominated Coteries

It has been shown that there exists a particular class of coteries that is particularly resilient to failures and network partitions [5, 6]; they are non-dominated coteries:

Definition 2.3 *Let C and D be two coteries under U. C* dominates *D iff $C \neq D$ and for each $Q_i \in D$ there is a $Q_j \in C$ such that $Q_j \subseteq Q_i$.* ■

So, a coterie, C is dominated iff there exists another coterie that dominates C. Otherwise C is ND (non-dominated) coterie. An ND coterie is superior to the dominated one since all partitions that survive under the dominated coterie can also survive under the ND one, the converse it is not true [6]. For example, the coterie $S = \{U\}$ (producing Lamport's algorithm [13]) is dominated by any singleton coterie. Examples of ND coteries are: majority coteries when considering n to be an odd number [6] and Maekawa's coteries [15].

2.3.2 Symmetric Coteries

There exist coteries in which the failure of one process makes unavailable all the quorums such as the ones formed by a single set of one element (singleton coteries: $C = \{\{p_i\}\}$). In truly distributed systems, it would be desirable that the impact of a failure of a process (in terms of quorums no longer available) be the same for all processes. In addition, since each process sends and receives a message from each process in the quorum it selected, each process try to select the smallest size quorum. Thus, it would be fair that quorums of a coterie be of the same size. The following definition is shown:

Definition 2.4 *A coterie C under U is* symmetric *iff the following properties hold [15]:*

- *(Equal Effort Property) For each $p_i \in U$ :: $|\{j | i \in Q_j\}| = \beta$.*
  i.e., all the processes are contained in the same number of quorums β.
- *(Equal Size Property) For each quorum Q_i :: $|Q_i| = \gamma$.*
  i.e., all quorum is the same size γ. ■

The coterie $S = \{U\}$ is symmetric with parameters $\beta = 1$ and $\gamma = n$. Other examples of symmetric coteries are: Majority coteries ($\beta = \binom{n-1}{\lfloor n/2 \rfloor}$ and $\gamma = \lfloor n/2 \rfloor + 1$), square grid coteries ($\beta = \gamma = 2\sqrt{n}$) [15] and Maekawa's coteries ($\beta = \gamma = O(\sqrt{n})$) [15]. The latter coteries meets the lower bound with respect to the quorum size of symmetric coteries.

2.4 k-Coteries

In order to solve the k mutual exclusion problem, Fujita et al. [10] and Huang et al.[9] have defined k-coteries. Dominated and non-dominated k-coteries have been discussed in [17]. Since Huang's definition has a case in which k processes cannot enter the critical section at the same time [16], we restrict our presentation to Fujita's definition [10].

Definition 2.5 *A set of quorums C is a* k-coterie *under U iff the following properties hold:*

- *(Non-intersection Property) For any $h(< k)$ distinct quorums $Q_1, Q_2, \ldots, Q_h \in C$ such that $Q_i \bigcap Q_j = \phi$ (for $1 \leq i \neq j \leq h$), there exists $Q \in C$ such that $Q \bigcap Q_i = \phi$ for i $(1 \leq i \leq h)$;*
- *(Intersection Property) For any $k+1$ quorums $Q_1, Q_2, \ldots, Q_{k+1} \in C$, there exists a pair i, j $(1 \leq j \neq i \leq k+1)$ such that $Q_i \bigcap Q_j \neq \phi$;*
- *(Minimality Property) For any pair of distinct quorums $Q_i, Q_j \in C :: Q_i \not\subseteq Q_j$.* ■

In a quorum-based algorithm for k-mutual exclusion, a process must wait an explicit permission, by means of a message, from each process of a quorum of a k-coterie before accessing the resource [11]. k-mutual exclusion is guaranteed since a k-coterie contains k pairwise disjoint quorums, but no more.

In [10], k-majority coteries and k-singleton coteries have been introduced as extensions of singleton and majority coteries. Recently, in [25] it has been shown that a k-majority is not necessarily a k-coteries. Although the k-coterie concept is a good tool to cope with k-mutual exclusion, its use cannot ensure safety in the h-out of-k mutual exclusion as shown in next Sections.

3 A Distributed h-out of-k Algorithm

3.1 Principle of the Algorithm

A simple approach to solve the h-out of-k mutual exclusion problem could be to reduce it either to a simple or to a k-mutual exclusion problem and resolve it using coteries or k-coteries. In both cases, to get its h units of the resource, a process needs to do h different requests. This situation may generate a deadlock during the acquisition phase of the units of the resource. For example, let us assume p_i requests h_i units and p_j requests h_j such that $h_i + h_j > k$. After a while, the following situtation could occur: p_i has obtained $h_x < h_i$ units, p_j has obtained $h_w < h_j$ units and $h_x + h_w = k$. This state deadlocks the algorithm. Hence, such an approach needs extra messages and corresponding message handlers to avoid deadlocks.

A more appropriate approach is to require in a single request all the h units of the resource and wait till all of them are available such as in simple and k-mutual exclusion. Thus, we derive the outline of a quorum-based h-out of-k mutual exclusion algorithm by extending Maekawa's algorithm [15] [1]. Using such an approach, each process may act both as a requesting process and as an arbiter in order to resolve conflicts for the acquisition of the units of the resource. The source code is split in two main parts: the first part is executed when a process, p_i, requests some units of the resource; it consists of the entry protocol and

[1] Note that k-mutual exclusion algorithms [10] using k-coteries were derived by extending Maekawa's algorithm.

that of exit from its critical section. The second part is composed of two message handlers: one for *request* messages and the other for *release* messages. A message handler is executed as soon as the corresponding message is received.

entry code: process p_i selects a quorum Q and sends a *request* message for h units of the resource to each member of Q and waits for the reception of the *permission* message from each process in Q;

critical section;

exit code: p_i sends a *release* message for h units of the resource to each member of Q.

upon receipt a request message from p_k:

if

$$\sum_{j \in U} c_j + h \leq k \tag{2}$$

where h is the number of units required from p_k and c_j is the number of units of the resource currently used by p_j from the point of view of process p_i (i.e., processes at which process p_i sent a permission message to and has not received a release message from).

then p_i sends back a *permission* to p_k and c_k is set to h;

else p_k's request is inserted into a waiting list.

upon receipt a release message from p_k: c_k is set to 0 and p_i tries to send *permission* messages to some requests in the waiting list using the relation 2.

Note that in the algorithm of the previous Section each request needs at least $3|Q|$ messages to be served: a round of *request* messages, a round of *permission* messages and one of *release* messages.

3.2 Correctness Proof

The definition of a set of quorums which can be assigned to each process when requesting some units of the resource, plays a key role in the detection of all conflicts and then in proving the safety property.

Theorem 3.1 *The algorithm in Section 3.1 guarantees safety if and only if*

$$\left(\forall \{p_{i_1}, p_{i_2}, \ldots, p_{i_{k+1}}\} \subseteq U \quad :: \bigcap_{i \in \{i_1, i_2, \ldots, i_{k+1}\}} Q_i \neq \emptyset \right)$$

where Q_i represents the quorum currently selected by process p_i to allocate the units of the resource.

Sufficiency. We prove that, if any set of $(k+1)$ quorums, $Q_1, Q_2, \ldots, Q_{k+1}$ satisfies

$$\bigcap_{1 \leq i \leq k+1} Q_i \neq \emptyset$$

then, safety is guaranteed. Assume that safety is not guaranteed. Thus, there is a set of requesting processes $R = \{p_{i_1}, p_{i_2}, \ldots, p_{i_t}\}$ which satisfies $\sum_{p \in R} c_p > k$ and every process has allocated its units. If $|R| > k+1$, select a subset R' which satisfies $|R'| = k+1$ and $\sum_{p \in R'} c_p > k$. If $|R| \leq k+1$, let $R' = R$. Let $R' = p_{i_1}, p_{i_2}, \ldots, p_{i_s}$ and Q_j be the quorum p_{i_j} has selected. Since $s \leq k+1$, $\bigcap_{1 \leq i \leq s} Q_i \neq \emptyset$. Let $p \in \bigcap_{1 \leq i \leq s} Q_i$. Since every request is accepted, p has sent a permission message to every request. Since $\sum_{p \in R'} c_p > k$, it contradicts the relation 2 of the algorithm.
Necessity. We prove that, if there exists a $(k+1)$ set of quorums $\{Q_1, \ldots, Q_{k+1}\}$ which satisfies

$$\bigcap_{i \in \{i_1, i_2, \ldots, i_{k+1}\}} Q_i = \emptyset$$

then safety is not guaranteed. Let us suppose p_i $(1 \leq i \leq k+1)$ requests one unit of the resource and uses Q_i as its quorum. Since $\bigcap_{i \in \{i_1, i_2, \ldots, i_{k+1}\}} Q_i = \emptyset$, no process in $\bigcup_{i \in \{i_1, i_2, \ldots, i_{k+1}\}} Q_i$ receives all requests. Thus, every process receives no more than k requests and sends a permission message to every request. Thus, every requesting process gets the unit of the resource, violating safety. ■

Let us spend some word about the liveness property (i.e., all the requests are eventually satisfied), which requires no occurrence of deadlocks or starvations. In the algorithm proposed in this Section, starvation can occur by applying, in each process, a deterministic rule which resolves conflicts always against some particular subset of processes (e.g. a rule based on the lexicographic order of process identifiers). On the contrary, a rule for conflict resolution based on timestamps [13] of requests does not produce starvation. Moreover, deadlock-freeness could also be violated since, due to the asynchrony of the communication networks (transmission times are unpredictable but finite), processes can receive a set of requests in different orders. Thus, they will send permission messages to different processes deadlocking actually the algorithm; i.e., no requesting process ever will get all permissions from its quorum. Hence, actions to prevent deadlocks are needed. Such actions increases the number of messages echanged per request that in the worst case is $5|Q|$ [2, 16]. The study of such actions (extra round of messages and corresponding message handlers) is out of the aim of the paper. The interested reader can refer to [2, 3, 21, 16].

4 k-Arbiters

4.1 Definition

Let us assume that processes select quorums from a set C; then Theorem 3.1 states that any $k+1$ quorums contained in C must be mutually intersecting. As k is greater than one, this condition cannot be satisfied either from coteries or from k-coteries. Indeed, coteries requires non-empty mutual intersection only between any *pair* of quorums apart from the value of k and k-coteries requires non-empty mutual intersection only between one pair of quorums contained in each distinct set of $k+1$ quorums. Thanks to Theorem 3.1 we are then able to define a new data structure based on a collection of quorums, called k-arbiter, which can be used to solve h-out of-k mutual exclusion problem.

Definition 4.1 *A set of quorums C is a* k-arbiter *under U iff the following properties hold:*

- *(Intersection Property) For any $k+1$ quorums $Q_1, Q_2, \ldots, Q_{k+1} \in C$, $\bigcap_{i=1}^{k+1} Q_i \neq \phi$;*
- *(Minimality Property) For any pair of distinct quorums $Q_i, Q_j \in C :: Q_i \not\subseteq Q_j$.* ■

Note that the k-arbiter becomes the coterie of Definition 2.2 when k is equal to one. In the following Sections, several examples of k arbiters are presented and classes of *non-dominated* and *symmetric* k-arbiters are investigated. Both classes hold the property of reliability against network partitions (non-dominated k-arbiter) and equality in load balancing (symmetric k-arbiter) described in Sections 2.3.1 and 2.3.2 respectively. Finally, a metric to measure the resiliency with respect to failures of k-arbiters is proposed.

4.2 Examples of k-Arbiters

4.2.1 Singleton k-Arbiters

A set C of quorums such that:

$$C = \{\{p_i\}\}$$

is called *singleton*. For this set of quorums the following property, whose proof is trivial and therfore it is omitted, follows:

Property 4.1 *A singleton set C is a k-arbiter.* ■

The singleton k-arbiter C corresponds to the case of the centralized h-out of-k mutual exclusion algorithm where a process detects actually all conflicts.

4.2.2 Uniform k-Arbiters

A set of quorums C such that:

$$C = \{Q \subset U |\ |Q| = \lfloor k \cdot n/(k+1) \rfloor + 1\}$$

is called *uniform*. For this set of quorums the following property holds:

Property 4.2 *A uniform arbiter C is a k-arbiter.*

Proof. For any quorum $Q_i \in C$ we have $|Q_i| = \lfloor k \cdot n/(k+1) \rfloor + 1$. Hence, for any $k+1$ quorums $Q_1, Q_2, \ldots, Q_{k+1} \in C$ we have $|Q_1 \bigcap Q_2 \bigcap \ldots \bigcap Q_{k+1}| \geq n - (k+1) \cdot (n - (\lfloor k \cdot n/(k+1) \rfloor + 1)) \geq 1$. Therefore the intersection property holds.
Minimality holds since for any pair $Q, Q' \in C$ such that $(Q \neq Q')$ we have $|Q| = |Q'|$. ■

Note that if $k = 1$, the uniform k-arbiter becomes the majority coterie (see Section 2.3).

4.2.3 $(k+1)$-Cube k-Arbiters

For the sake of simplicity we initially suppose $n = a^{k+1}$ for some integer a. Quorums of a $(k+1)$-cube set are easily defined by means of a geometric argument: each process is assigned to a different point x in the $(k+1)$-dimensional space $\mathbb{E}^{k+1}$ having integer coordinates in the range $[0,\ \sqrt[k+1]{n} - 1]$. Each quorum is associated to point x and consists in the set of processes whose corresponding point has at least one coordinate value equal to the corresponding coordinate of x. In other words, a quorum associated with point x is defined by all processes whose corresponding point is contained in at least one of the $k+1$ isothetic hyperplanes (one for each coordinate) passing through x. With $x = (i_1, i_2, \ldots, i_{k+1})$ we have:

$$Q_{i_1,\ldots,i_{k+1}} = \{(x_1, x_2, \ldots, x_{k+1}) | x_1 = i_1\} \cup \{(x_1, x_2, \ldots, x_{k+1}) | x_2 = i_2\} \cup \ldots \ldots \cup \{(x_1, x_2, \ldots, x_k, x_{k+1}) | x_k = i_k\} \cup \{(x_1, x_2, \ldots, x_{k+1}) | x_{k+1} = i_{k+1}\}$$

It can be shown that the size of each quorum is:

$$Q_i = n - (\sqrt[k+1]{n} - 1)^{k+1} = \sum_{i=0}^{k} (-1)^{i+1} \binom{k+1}{i} n^{\frac{k-i}{k+1}} \leq (k+1) n^{\frac{k}{k+1}}$$

For quorums of the $(k+1)$-cube set, the following property holds:

Property 4.3 *In the case $n = a^{k+1}$ for some integer a, a $(k+1)$-cube set C is a k-arbiter.*

Proof. For any $(k+1)$ quorums $Q_{i_1^1, i_2^1, \ldots, i_{k+1}^1}, Q_{i_1^2, i_2^2, \ldots, i_{k+1}^2}, \ldots Q_{i_1^{k+1}, i_2^{k+1}, \ldots, i_{k+1}^{k+1}} \in C$ we have process $(i_1^1, i_2^2, \ldots, i_{k+1}^{k+1})$ is contained in every quorum. Thus, the intersection property holds.
Minimality holds since for any pair $Q, Q' \in C$ such that $(Q \neq Q')$ we have $|Q| = |Q'|$. ■

Note that the $(k+1)$-cube k-arbiter becomes the square grid coterie (see Section 2.3) for the simple mutual exclusion. If $n \neq a^{k+1}$, some care is needed while assigning points in $\mathbb{E}^{k+1}$

to processes. It is sufficient to assign processes to points according to the lexicographic order of points' coordinates. Thus we will assign process p_1 to point $(0,\ldots,0)$, p_2 to $(0,\ldots,0,1)$, p_3 to $(0,\ldots,0,2)$, $\ldots$, $p_{\lceil \sqrt[k+1]{n} \rceil}$ to $(0,\ldots,0,\lceil \sqrt[k+1]{n} \rceil)$, $p_{\lceil \sqrt[k+1]{n} \rceil+1}$ to $(0,\ldots,1,0)$. In short, the coordinates of the point associated with process p_i are simply obtained by representing integer i in base $\lceil \sqrt[k+1]{n} \rceil+1$. In this way there will be exactly $\left\lfloor n/\lceil \sqrt[k+1]{n} \rceil^k \right\rfloor$ filled hyperplanes $X_1 = v$, where $v = 0, 1, \ldots$, and at most one hyperplane will be partially filled (more precisely hyperplane $X_1 = \left\lfloor n/\lceil \sqrt[k+1]{n} \rceil^k \right\rfloor + 1$). Quorum Q_i is then defined, as for the simple case, by the $k+1$ isothetic hyperplanes passing through point x_i.

Theorem 4.4 *In the general case, a $(k+1)$-cube set C is a k-arbiter.*

Proof. As before we have to prove that for any $(k+1)$-ple of qourums in C an arbiter exists. The problem now is that a point represents an arbiter only if it is assigned to a process. Fix a set S of $k+1$ quorums, and let L be the number of non-empty hyperplanes $X_1 = v$ in our arrangement: if all processes in S lie in hyperplane $X_1 = L-1$ then any process in S is an arbiter itself for S. Otherwise we can always find a point $(x_1, x_2, \ldots, x_{k+1})$ that represents an arbiter by choosing x_1 as one of the coordinate values for X_1 in S which are less than $L-1$. This point is in a full hyperplane $X_1 = x_1$, hence values for the remaining coordinates can be arbitrarily chosen from points associated to the remaining processes in S. ■

4.3 Non-dominated k-Arbiters

We can introduce the notion of non-dominated k-arbiters as follows:

Definition 4.2 *Let C and D two k-arbiters under U. C* dominates *D iff $C \neq D$ and for each $Q_i \in D$ there is a $Q_j \in C$ such that $Q_j \subseteq Q_i$.* ■

A k-arbiter, C is then dominated iff there exists a k-arbiter that dominates C. Otherwise C is an ND (non-dominated) k-arbiter. If k-arbiter D dominates C, D survives to all failures and network partitions at which D survives. The following theorem is usefull to check if a k-arbiter is ND:

Theorem 4.5 *Let C be a k-arbiter under U, C is dominated iff there exists a quorum H such that*

- *For any $Q \in C$ :: $H \not\supseteq Q$;*
- *For any k quorums $Q_1, Q_2, \ldots, Q_k \in C$ :: $H \cap \left(\bigcap_{j=1}^{k} Q_j\right) \neq \phi$.*

Proof. The proof is similar to the one proposed in [17] that proves under which conditions a k-coterie is a non-dominated one. Let C be a k-arbiter under U. First, we show that if C is dominated, then above properties are satisfied. Suppose C is dominated by $D(\neq C)$. There are two cases:

1. $C \subset D$
   Any set $H \in D - C$ satisfies both properties (otherwise D is not a k-arbiter);

2. $C \not\subseteq D$
   There is a quorum $Q' \in C$ and $H \in D$ such that $H \subset Q'$. Now we show that H satisfies the above properties:

   Assume that H does not satisfy the first property. Then, $Q \subseteq H$ for some $Q \in C$. Since $Q \subseteq H$ and $H \subset Q'$, $Q \subset Q'$. This contradicts the minimarity condition of arbiter C. Thus, the first property is satisfied;

   Assume that H does not satisfy the second property. Then, for some k quorum $Q_1, Q_2, \ldots, Q_k \in C$, $H \cap \left(\bigcap_{j=1}^{k} Q_j\right) = \phi$. Since C is dominated by D, for each $Q_i (1 \leq i \leq k)$, there is a quorum $H_i \in D$ such that $H_i \subseteq Q_i$. These quorums satisfy $H \cap \left(\bigcap_{j=1}^{k} H_j\right) = \phi$ and this violates the intersection property for D. Thus, the second property is satisfied.

   Now, we show that if both properties are satisfied for some H, C is dominated. There are two cases:

   (a) $H \subset Q$ for some $Q \in C$
   Let $C' = \{Q \in C | H \subseteq Q\}$ and $D = C - C' \cup \{H\}$. It is easy to check that D is a k-arbiter which dominates C;

   (b) $H \not\subset Q$ for all $Q \in C$
   Let $D = C \cup \{H\}$. It is easy to check that D is a k-arbiter which dominates C. ■

The singleton k-arbiter is an ND k-arbiter. The following theorem gives the case that uniform k-arbiter is non-dominated.

Theorem 4.6 *When $n = c(k+1)+1$ for some integer c, uniform k-arbiter is non-dominated.*

Proof. Assume that uniform k-arbiter C is dominated. By Theorem 4.5, there exist a set $H \subseteq U$ which satisfies the two properties. Since $Q \not\subseteq H$ for all $Q \in C$ and C is all subset of U with size $\lfloor k \cdot n/(k+1) \rfloor + 1 = k \cdot c + 1$, $|H| \leq k \cdot c$. Without loss of generality, let $H = \{p_1, p_2, \ldots, p_{|H|}\}$. Now select k quorum $Q_1, Q_2, \ldots Q_k \in C$ as $Q_i = U - \{p_{(i-1)c+1}, p_{(i-1)c+2} \cdots, p_{i \cdot c-1}, p_{i \cdot c}\}$ $(1 \leq i \leq k)$. It is easy to see that $H \cap \left(\bigcap_{j=1}^{k} Q_j\right) = \phi$. This is a contradiction. Therefore, C is non-dominated. ■

Theorem 4.6 extends the result obtained in [6] concerning domination of majority coteries. Infact, if k is equal to one, uniform k-arbiters boil down to majority coteries that are non-dominated when considering n to be an odd number.

Finally, $k+1$-cube k-arbiters are dominated, to prove that we show a simple counter-example when considering k equal to one and n to 4. In this case we have the following 1-arbiter (i.e., the square grid coterie):

$$C = \{\{1,2,3\},\ \{1,2,4\},\ \{1,3,4\},\ \{2,3,4\}\}$$

that is dominated by the following 1-arbiter:

$$Q = \{\{1,2,3\},\ \{1,4\},\ \{2,3,4\}\}$$

Note that the 1-arbiter C coincides actually with the uniform 1-arbiter for a system of 4 processes that is dominated from Theorem 4.6. Examples of dominated $k+1$-cube k-arbiters with $k > 1$ (for example $k = 2$ and $n = 8$) are left to the reader.

4.4 Symmetric k-Arbiters

As for coteries, we can introduce the concept of *symmetric* k-arbiters.

Definition 4.3 *A k-arbiter C under U is symmetric iff the following properties hold:*

- *(Equal Effort Property) For each $p_i \in U$ :: $|\{j | i \in Q_j\}| = \beta$.*
  i.e., all the processes are contained in the same number of quorums β.
- *(Equal Size Property) For each quorum Q_i :: $|Q_i| = \gamma$.*
  i.e., all quorum is the same size γ. ■

Considering the previous examples of k-arbiters, it is trivial to see that a singleton k-arbiter is non-symmetric and the failure of one process may make it unavailable. The uniform k-arbiter is a *symmetric* k-arbiter. Indeed, with simple combinatorial arguments, we get $\beta = \binom{n-1}{\lfloor k \cdot n/(k+1) \rfloor}$ and, by definition, $\gamma = \lfloor k \cdot n/(k+1) \rfloor + 1$. Finally, when $n = a^{k+1}$, the $(k+1)$-cube is a *symmetric* k-arbiter and with the same arguments used to define the size of each quorum we have $\beta = \gamma = (k+1)n^{\frac{k}{k+1}}$.

Lower Bound of Symmetric k-Arbiters

It is interesting to show a lower bound for the maximum size of quorums $|Q|$ of a symmetric k-arbiter. Let n be the number of processes, $\gamma = |Q|$ be the quorum size, and y be the number of quorums in D. Suppose that every process is contained in β quorums. By counting the sum of the sizes of quorums in two ways, the equality $n \cdot \beta = \gamma \cdot y$ is obtained. The number of different combinations of $(k+1)$ quorum sets from y quorums is C^y_{k+1}. Since each process occurs in β quorums, each process is in the intersection of $(k+1)$ quorums for C^β_{k+1} different quorum combinations. Thus, $C^y_{k+1} \leq n \cdot C^\beta_{k+1}$ must hold. This inequality can be written as $y(y-1)\cdots(y-k) \leq n \cdot \beta(\beta-1)\cdots(\beta-k)$.

From the above equality and inequality, $\gamma > n^{k/k+1}$ is obtained.

It follows that the last inequality is a lower bound for $|Q|$. This bound is not new, it was found both studying families of sets called *t-wise s-intersecting* ($t \geq 2$ and $s \geq 1$)[4] and the weighted distributed match-making problem [12]. It is simple to show that our problem is equivalent to theirs. For example, a k-arbiter is actually a *(k+1)-wise 1-intersecting* family of subsets of U. In [4] it has been proved that a *(k+1)-wise 1-intersecting* family is equivalent to a finite projective space of dimension $k+1$. In that discrete space, quorums correspond to hyperplanes and point to processes. From the properties of a finite projective space, the above bound follows. Conditions for which a finite projective space exits are given in [4]. Note that, if $k = 1$, the lower bound becomes $|Q| > n^{1/2}$ that is Maekawa's lower bound for symmetric coteries, that was proved using the theory of finite projective planes [15].

4.5 A Measure to Evaluate Resiliency with respect to Failures

As suggested in [7], given the high number of choices in selecting the "best" coterie or k-arbiter for a particular distributed system of n processes, we need a set of metrics to evaluate the "reliability" provided by each choice. In this section we propose a metric that can be used, toghether with the ones proposed in [7], to select a k-arbiter.

Each time a process p_i fails, all quorums of the k-arbiter containing p_i are actually no longer available. This suggest to introduce a metrics that evaluates the *the maximun impact of one process failure on a k-arbiter* in terms of its quorum availability. In other words, this measure gives an idea on how much the failure of one process influences the quorum availability of the k-arbiter apart from analysis that consider the probability that a process is (or not) in an operational state as the one proposed in [6, 7].

Let $\mathcal{A}$ be a k-arbiter, let $n_{\mathcal{A}}$ be the number of all the quorums in $\mathcal{A}$ and let $\beta^i_{\mathcal{A}}$ be the number of quorums in $\mathcal{A}$ containing process p_i. The resiliency to failure of the k-arbiter $\mathcal{A}$, $\mathcal{R}_{\mathcal{A}}$ is measured by the following ratio:

$$\mathcal{R}_{\mathcal{A}} \;=\; \mathbf{max}\left(\frac{\beta^i_{\mathcal{A}}}{n_{\mathcal{A}}}\right)_{i=1,\ldots,n}$$

If $\mathcal{R}_{\mathcal{A}}$ is equal to one then there exists at least one process which makes unavailable all the quorums of $\mathcal{A}$, if it fails. Examples of such a k-arbiter are the singleton one and all the k-arbiters dominated by any singleton one (e.g. $\{\{U\}\}$). The smaller $\mathcal{R}_{\mathcal{A}}$ is, the more reduced is the impact of the failure of one process on quorum availability. Ideally, if $\mathcal{R}_{\mathcal{A}}$ was equal to zero, $\mathcal{A}$ would be uninfluenced by failures. Concerning, k-arbiters we have introduced in the previous Section, for the uniform we have:

$$\mathcal{R}_{uni} = \frac{\binom{n-1}{\lfloor k\cdot n/(k+1)\rfloor}}{\binom{n}{\lfloor k\cdot n/(k+1)\rfloor+1}} \leq \frac{k}{k+1} + \frac{1}{n}$$

For the $k+1$-cube k-arbiter, in the case $n = a^{k+1}$ for some integer a, we have:

$$\mathcal{R}_{cube} = \frac{(k+1)N^{\frac{k}{k+1}}}{n} = \frac{k}{n} + \frac{1}{n}$$

In the case of k-arbiters that match the lower bound for symmetric k-arbiters, we have:

$$\mathcal{R}_{lbs} = \frac{N^{\frac{k}{k+1}}}{n} = \frac{1}{n}$$

Hence, assuming $n \geq k+1$, we have the following inequality:

$$0 < \mathcal{R}_{lbs} < \mathcal{R}_{cube} \leq \mathcal{R}_{uni} < \mathcal{R}_{sing} = 1$$

5 Conclusion

Simple and k-mutual exclusion do not capture cases in which the amount of shared resource might differ from request to request. In this paper we have formalized this problem as the *h-out of-k mutual exclusion problem*, in which each request concerns some number h ($1 \leq h \leq k$) of units of shared resource and no unit is allocated to multiple processes at the same time. We have shown a general scheme for a quorum-based h-out of-k mutual exclusion algorithm that relies on a collection of quorums called *k-arbiter* which subsumes the concept of coterie. Several examples of k-arbiters, that extend well-known types of coteries, have been introduced; two particularly interesting classes of k-arbiters have been investigated and a metric to evaluate the resiliency with respect to failures of k-arbites has also been proposed. As a future work, it would be interesting to find a k-arbiter that subsumes *tree coteries*, introduced in [1], in order to get logarithmic quorum size maintaining high quorum availability.

References

[1] Agrawal, D. and El-Abbadi, A.: "An Efficient and Fault-tolerant Solution for Distributed Mutual Exclusion," *ACM Transactions on Computer Systems*, Vol. 9, No. 1, pp. 1–20 (1991).

[2] Baldoni, R.: "An $O(N^{\frac{M}{M+1}})$ Distributed Algorithm for the k-out of-M Resource Allocation Problem" *Proc. of 14-th IEEE Int. Conf. on Distributed Computing Systems*, pp. 81–87 (1994).

[3] Chandy, K.M. and Misra, J.: "The Drinking Philosopher Problem," *ACM Transactions on Programming Language and Systems*, Vol. 6, No. 4, pp. 632–642 (1982).

[4] P. Frankl and Z. Furedi: "Finite Projective Spaces and Intersecting Hypergraphs," *Combinatorica*, Vol. 6, No. 4, pp. 335–354 (1986).

[5] Garcia-Molina, H. and Barbara, D.: "How to Assign Votes in a Distributed System," *Journal of the ACM*, Vol. 32, No. 4, pp. 841–860 (1985).

[6] Garcia-Molina, H. and Barbara, D.: "The Reliability of Voting Mechanism," *IEEE Transactions on Computers*, Vol. C-36, No. 10, pp. 1197–1208 (1987).

[7] Garcia-Molina, H. and Barbara, D.: "Mutual Exclusion in Partitionated Distributed Systems," *Distributed Computing*, Vol. 1, pp. 119–132 (1986).

[8] Helary, J.M., Mostefaoui A. and Raynal, M.: "A General Scheme for Token- and Tree-based Distributed Mutual Exclusion Alfgorithms," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 11, pp. 1185–1196. .

[9] Huang, S. T., Jiang, J. R., and Kuo, Y. C.: "k-Coteries for Fault-Tolerant k Entries to a Critical Section," *Proc. of 13-th IEEE Int. Conf. on Distributed Computing Systems*, pp. 74–81 (1993).

[10] Kakugawa, H., Fujita, S., Yamashita, M., and Ae, T.: "Availability of k-Coterie," *IEEE Transactions on Computers*, Vol. 42, No. 5, pp. 553–558 (1993).

[11] Kakugawa, H., Fujita, S., Yamashita, M., and Ae, T.: "A Distributed k-Mutual Exclusion Algorithm using k-Coterie," *Information Processing Letters*, Vol. 49, pp. 213–238 (1994).

[12] E. Kranakis and P. Vitanyi: "A Note on Weighted Distributed Match Making, " *Mathematical System Theory*, Vol. 25, pp. 123–140 (1992).

[13] Lamport, L.: "Time, Clocks, and the Ordering of Events in a Distributed System," *Communications of ACM*, Vol. 21, No. 7, pp. 558–565 (1978).

[14] Lamport, L.: "The Implementation of Reliable Distributed Multiprocessor Systems," *Computer Networks*, Vol. 2, pp. 95–114 (1978).

[15] Maekawa, M.: "A $\sqrt{N}$ Algorithm for Mutual Exclusion in Decentralized Systems," *ACM Transactions on Computer Systems*, Vol. 3, No. 2, pp. 145–159 (1985).

[16] Manabe, Y. and Aoyagi, S.: "A Distributed k-Mutual Exclusion Algorithm Using k-Coterie," *IEICE Technical Report* COMP93–43 (Sept. 1993).

[17] Neilsen, M.L. and Mizuno, M.: "Nondominated k-coteries for Multiple Mutual Exclusion," *Information Processing Letters*, Vol. 50, pp. 247–252 (1994).

[18] Raymond, K.: "A Distributed Algorithm for Multiple Entries to a Critical Section," *Information Processing Letters*, Vol. 30, No. 4, pp. 189–193 (1989).

[19] Raynal, M.: "A Distributed Solution for the k-out of-m Resources allocation problem," *Lecture Notes in Computer Sciences*, Springer Verlag, Vol. 497, pp. 599–609, (1991).

[20] Raynal, M.: "A Simple Taxonomy for Distributed Mutual Exclusion," *ACM Operating System Review*, Vol. 25, No. 1, pp. 47–51 (1991).

[21] Raynal, M.: "Synchronization and global states in Distributed System," *Ed. Eyrolles*, 200 pages (1992).

[22] R.D. Schlichting, F.B., Schneider: "Fail-stop processors: an approach to designing fault-tolerant computing systems," *ACM Trans. on Computing Systems*, Vol. 1, no. 3, pp. 222-238 (1983).

[23] Singhal, M.: "A Taxonomy of Distributed Mutual Exclusion," *Journal of Parallel and Distributed Computing*, Vol. 18, No. 1, pp. 94-101 (1993).

[24] Thomas, R.: "A Majority consensus approach to concurrency control for multiple copy databases," *ACM Transactions on Database Systems*, Vol. 4, No. 2, pp. 180–209 (1979).

[25] Yuan, S. M. and Chang, H.K.: "Comments on: Availability of k-coterie," *IEEE Transactions on Computerss*, Vol. 43, No. 12, pp. 1457 (1994).

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399