J. Parallel Distrib. Comput. 77 (2015) 95–104

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Mutual inclusion in asynchronous message-passing distributed systems

Hirotsugu Kakugawa

Graduate School of Information Science and Technology, Osaka University, 1-5 Yamadaoka, Suita, Osaka, 565-0871, Japan

HIGHLIGHTS

- The mutual inclusion problem is a problem such that at least one process is in critical section.
- This paper proposes two distributed solutions in the asynchronous message passing model.
- We discuss the relation between mutual inclusion and mutual exclusion.

ARTICLE INFO

Article history:
Received 29 November 2013
Received in revised form
17 October 2014
Accepted 2 January 2015
Available online 8 January 2015

Keywords:
Distributed algorithm
Mutual exclusion mutual inclusion
Process synchronization

ABSTRACT

In the mutual inclusion problem, at least one process is in the critical section. However, only a solution for two processes with semaphores has been reported previously. In this study, a generalized problem setting is formalized and two distributed solutions are proposed based on an asynchronous message-passing model. In the local problem setting (the local mutual inclusion problem), for each process P, at least one of P and its neighbors must be in the critical section. For the local problem setting, a solution is proposed with $O(\Delta)$ message complexity, where Δ is the maximum degree (number of neighboring processes) of a network. In a global setting (the global mutual inclusion problem), at least one of the processes must be in the critical section. For the global problem setting, a solution is proposed with $O(|Q|)$ message complexity, where $|Q|$ is the maximum size for the quorum of a coterie used by the algorithm, which is typically $|Q| = \sqrt{n}$, where n is the number of processes in a network.

1. Introduction

This study considers a problem called distributed *mutual inclusion*, which is complementary to the distributed mutual exclusion problem. Informally, in the mutual inclusion problem (abbreviated to mutin problem), *at least one* process is in the critical section (abbreviated to CS), whereas *at most one* process is in the CS in the mutual exclusion problem (abbreviated to mutex problem). To the best of the author's knowledge, [10] is the only previous study of the mutin problem. In [10], a solution was proposed for only two processes with semaphores. The present study formalizes a generalized problem setting and two distributed solutions are proposed based on an asynchronous message-passing model.

After the first study [10] of mutual inclusion was published, the problem of mutual inclusion was ignored by the research community. This was mainly because there were no known practical applications of mutual inclusion. However, the applications of mutual inclusion now include the effective maintenance of clustering in sensor networks and the replacement of servers in server/client systems.

The main motivation for studying the distributed mutin problem is the theoretical formulation of handover for cluster heads in sensor networks. A cluster head process (node) often changes its role to become an ordinary process (and vice versa) while maintaining a clustering condition to equalize the energy consumption between processes. Suppose that a cluster head process is in the CS and that an ordinary process is in the non-CS. Mutual inclusion then maintains a clustering condition dynamically, *i.e.*, each process is a cluster head or at least one of its neighbor is a cluster head. The handover problem for cluster heads is formulated as a process synchronization problem in the present study.

In this study, generalized settings for the mutin problem are proposed, i.e., the ℓ-mutin problem, where at least ℓ processes are in the CS. Local and global settings are also proposed for the 1-mutin problem. In the local ℓ-mutin problem on network $G = (V, E)$, for each $P_i \in V$, at least ℓ processes among P_i and its neighbors are in the CS. The global ℓ-mutin problem is a special case of the local ℓ-mutin problem when $G = (V, E)$ is complete. As a first step, distributed solutions are proposed for the local and global 1-mutin problems. For the local mutin problem, a solution with

E-mail address: kakugawa@ist.osaka-u.ac.jp.

http://dx.doi.org/10.1016/j.jpdc.2015.01.003
0743-7315/

$O(\Delta)$ message complexity is proposed, where Δ is the maximum degree (number of neighboring processes) of a network. For the global mutin problem, a solution with $O(|Q|)$ message complexity is proposed, where $|Q|$ is the maximum size of the quorum of a coterie used by the algorithm, which is typically $|Q| = \sqrt{n}$, and n is the number of processes in a network.

The remainder of this paper is organized as follows. Section 2 provides several definitions and problem statements. Section 3 reviews related research and presents some observations on the relationships between the mutin and mutex problems. Section 4 provides a solution to the local 1-mutin problem. Section 5 gives a solution to the global 1-mutin problem. In Section 6, the relationship between the 1-mutin problem and the dominating set is discussed. Section 7 comprises a summary of this study and suggestions for future research.

2. Preliminary

Let $G = (V, E)$ be a graph, where $V = \{P_1, P_2, \ldots, P_n\}$ is a set of processes and $E \subseteq V \times V$ is a set of bidirectional communication links between a pair of processes. We assume that $(P_i, P_j) \in E$ if and only if $(P_j, P_i) \in E$. Each communication link is FIFO. We consider that G is a distributed system. The number of processes in $G = (V, E)$ is denoted by $n(= |V|)$. A set of neighbors of $P_i \in V$ is denoted by N_i, where $N_i = \{P_j \mid (P_i, P_j) \in E\}$. We assume that the distributed system is asynchronous, i.e., there is no global clock. A message is delivered eventually but there is no upper bound on the delay time and the running speed of a process may vary.

We assume that each process $P_i \in V$ maintains a variable $state_i \in \{\text{InCS}, \text{NonCS}\}$ and that the initial value of each $state_i$ is InCS. The behavior of each process P_i is as follows, where we assume that P_i eventually invokes Entry-Sequence when it is in the NonCS state.

```
/* InCS */
while true {
  Exit-Sequence;
  /* NonCS */
  Entry-Sequence;
  /* InCS */
}
```

Definition 1 (*The ℓ-mutin Problem*)**.** A protocol $\mathcal{P}$ solves the ℓ-mutin problem on network $G = (V, E)$, where $1 \leq \ell \leq n$, if and only if the following two conditions hold. Safety: For each process $P_i \in V$, at least ℓ of P_i or its neighbor $P_j \in N_i$ are in the InCS state at any time. Liveness: Each process $P_i \in V$ enters the NonCS and InCS states alternately infinitely often.

The ℓ-mutual inclusion problem requires the design of a protocol for performing the Exit-Sequence and Entry-Sequence. We refer to this problem as the *local* ℓ-mutin problem, which contrasts with the *global* ℓ-mutin problem, where the latter is a special case when G is a complete network, *i.e.*, $N_i = V \setminus \{P_i\}$ for each $P_i \in V$.

The typical performance measures applied to mutin algorithms are as follows.

- *Message complexity*: the number of message exchanges triggered by a pair of invocations of the Exit-Sequence and Entry-Sequence.
- *Concurrency*: the minimum number of processes that are in the CS simultaneously.
- *Waiting time*[1]: the time period between the invocation of the Exit-Sequence and completion of the exit from the CS.

[1] The name of this performance measure differs among previous studies and some (e.g., [33]) refer to this performance measure as the "synchronization delay".

3. Related work

The mutin problem was proposed in [10], which is the only previous study to address this problem. A closely related process synchronization problem is the mutex problem, which has been studied extensively. In this section, we first observe the relationship between the global mutin and global mutex problems, before reviewing previous studies of the mutex problem.

3.1. Relationship between mutual inclusion and mutual exclusion

In this subsection, we discuss the relationship between the global mutin and global mutex problems. There are similarities between the local mutin and local mutex problems, but we only consider the global version to simplify the results. Let $n \geq 1$ be integers. The global mutin and mutex algorithms discussed in this section are implicitly assumed to be algorithms on a network of size n.

Informally, the global j-mutin and global j-mutex problems are defined as follows. For each $0 \leq j \leq n$, where n is the total number of processes, the global j-mutin problem has at least j processes in the CS whereas the global j-mutex problem has at most j processes in the CS.

Let A be an algorithm for global j-mutin or global j-mutex for some $0 \leq j \leq n$. By Co-A, we denote a complemented algorithm of A, which is obtained by swapping the process states, InCS and NonCS.

We make the following simple observations.

Observation 1. *For each j such that $0 \leq j \leq n$ and for each global j-mutin or j-mutex algorithm A, Co-(Co-A) $= A$ holds.* □

Observation 2. *For each j such that $0 \leq j \leq n$ and for each global j-mutex algorithm A, Co-A is a global $(n-j)$-mutin algorithm.*

Proof. By A, at most j processes are in the InCS state. Hence, by Co-A, at most j processes are in the NonCS state, which implies that at least $n - j$ processes are in the InCS state. □

Observation 3. *For each j such that $0 \leq j \leq n$ and for each global j-mutin algorithm A, Co-A is a global $(n-j)$-mutex algorithm.*

Proof. By A, at least j processes are in the InCS state. Hence, by Co-A, at least j processes are in the NonCS state, which implies that at most $n - j$ processes are in the InCS state. □

Hence, mutin and mutex are essentially the same problem, which is summarized as follows. For each j such that $0 \leq j \leq n$, let $\mathcal{I}_n^j$ be a set of global j-mutin algorithms, and $\mathcal{E}_n^j$ is a set of global j-mutex algorithms. Let Co-$\mathcal{I}_n^j = \{\text{Co-}A : A \in \mathcal{I}_n^j\}$, and Co-$\mathcal{E}_n^j = \{\text{Co-}A : A \in \mathcal{E}_n^j\}$.

Observation 4. *For each j such that $0 \leq j \leq n$, $\mathcal{I}_n^j =$ Co-$\mathcal{E}_n^{n-j}$ and $\mathcal{I}_n^{n-j} =$ Co-$\mathcal{E}_n^j$.* □

Let $\mathcal{I}_n = \cup_{0 \leq j \leq n} \mathcal{I}_n^j$, $\mathcal{E}_n = \cup_{0 \leq j \leq n} \mathcal{E}_n^j$, Co-$\mathcal{I}_n = \{\text{Co-}A : A \in \mathcal{I}_n\}$, and Co-$\mathcal{E}_n = \{\text{Co-}A : A \in \mathcal{E}_n\}$.

Observation 5. *$\mathcal{I}_n =$ Co-$\mathcal{E}_n$ and $\mathcal{E}_n =$ Co-$\mathcal{I}_n$.* □

Now, we make the following conclusions.

1. The complexity of the j-mutin problem is the same as the complexity of the $(n-j)$-mutex problem.
2. The family of mutin algorithms and the family of mutex algorithms are complementary.

A study of the distributed j-mutin problem may yield a new algorithmic framework for the distributed k-mutex problem.