Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Processing Letters 91 (2004) 85–91
Information Processing Letters
www.elsevier.com/locate/ipl

Speedup of Vidyasankar's algorithm for the group k-exclusion problem

Masataka Takamura [a,*], Tom Altman [b], Yoshihide Igarashi [a]

[a] *Department of Computer Science, Gunma University, Kiryu 376-8515, Japan*
[b] *Department of Computer Science and Engineering, University of Colorado at Denver, Denver, CO 80217, USA*

Received 3 September 2003; received in revised form 12 March 2004

Available online 6 May 2004

Communicated by M. Yamashita

Abstract

Vidyasankar introduced a combined problem of k-exclusion and group mutual exclusion, called the group k-exclusion problem, which occurs in a situation where philosophers with the same interest can attend a forum in a meeting room, and up to k meeting rooms are available. We propose an improvement to Vidyasankar's algorithm. Waiting times in the trying region in the original algorithm and in our algorithm are bounded by $n(n-k)c + O(n^3(n-k))l$ and $(n-k)c + O(n(n-k)^2)l$, respectively, where n is the number of processes, l is an upper bound on the time between successive two atomic steps, and c is an upper bound on the time that any philosopher spends in a forum.

Keywords: Distributed computing; Group mutual exclusion; k-Exclusion; Lockout avoidance; Mutual exclusion; Shared memory

1. Introduction

The k-exclusion problem is a natural generalization of the mutual exclusion problem. In k-exclusion, k users are allowed to use the resource concurrently [1,5]. Group mutual exclusion is another natural generalization of mutual exclusion [3]. It is required in a situation where a resource can be shared by processes of the same group, but not by processes of different groups [3,7]. The group mutual exclusion can be described as the *congenial talking philosophers* problem, where the philosophers can spend their time thinking alone, but when a philosopher is tired of thinking, he/she attempts to attend a forum and to talk at the forum. For the congenial talking philosophers problem, we assume that there is only one meeting room.

A further generalization of the mutual exclusion problem, called the group k-exclusion problem, was introduced by Vidyasankar in [8,9]. He proposed a simple algorithm in the asynchronous multi-writer/reader shared memory

* Corresponding author.
E-mail addresses: takamura@comp.cs.gunma-u.ac.jp (M. Takamura), taltman@carbon.cudenver.edu (T. Altman), zeneiig@comp.cs.gunma-u.ac.jp (Y. Igarashi).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2004.03.012

model for the problem by incorporating the n-process algorithm by Peterson [6], but did not analyze the waiting time [9].

In this paper, we first analyze Vidyasankar's algorithm to estimate an upper bound on the waiting time. Then we show that we can accelerate his algorithm by incorporating techniques used in [2,5]. Upper bounds on waiting times in the trying region by Vidyasankar's algorithm and by our algorithm are $n(n-k)c + O(n^3(n-k))l$ and $(n-k)c + O(n(n-k)^2)l$, respectively, even if there exist at most $k-1$ faulty processes of the stopping type, where n is the number of philosophers (also the number of processes), l is an upper bound on the time between successive two atomic steps, and c is an upper bound on the time that any philosopher spends in a forum.

2. Preliminaries

The computational model used in this paper is the asynchronous shared memory model as given in [4]. It is a collection of processes and shared variables. Processes take steps at arbitrary speeds, and there is no global clock. Interactions between a process and its corresponding philosopher are by input actions from the philosopher to the process and by output actions from the process to the philosopher. All communication among the processes is via shared memory. A shared variable is said to be atomic if it has property that read operations and write operations behave as if they occur in some total order. In this paper, we assume that all shared variables are atomic.

A philosopher with access to a forum is modeled as being in the talking region. When a philosopher is not involved in any forum, he/she is said to be in the thinking region. In order to gain admittance to the talking region, his/her corresponding process executes a trying protocol. The duration from the start of execution of the trying protocol to the entrance into the talking region is called the trying region. After the end of talking by a philosopher at a forum in a meeting room, his/her corresponding process executes an exit protocol. The duration of execution of the exit protocol is called the exit region. These regions are followed in cyclic order, from the thinking region to the trying region, to the talking region, to the exit region, and then back again to the thinking region. The group k-exclusion problem is to devise protocols for the philosophers to efficiently and fairly attend a forum when they wish to talk, under the conditions that there are k meeting rooms and that at most one forum can be held in any meeting room at the same time.

We assume n philosophers, $P_1, \ldots, P_n$, who want to spend their time either thinking alone or talking in a forum. We also assume that there are m different fora. Each P_i $(1 \leqslant i \leqslant n)$ corresponds to process i. The inputs to process i from P_i are $try_i(f)$ which means a request by P_i for access to forum $f \in \{1, \ldots, m\}$ to talk there, and $exit_i$ which means an announcement of the end of talking by P_i. The outputs from process i to P_i are $talk_i$ which means granting attendance at a requested forum to P_i, and $think_i$ which means that P_i can continue with his/her thinking alone. We assume that a philosopher in a forum spends an unpredictable but a finite length of time there. The system to solve the group k-exclusion problem should satisfy the following conditions (G1, G2, G3):

(G1) *Group k-exclusion*: At any time, the number of different fora held in meeting rooms is at most k, and different fora cannot be held in the same meeting room at the same time.
(G2) *k-lockout avoidance*: If any faultless process in the trying region always leaves after spending a finite length of time there, and if the number of faulty process of the stopping type is at most $k-1$, then any faultless process wishing to enter the talking region eventually does so.
(G3) *Progress for the exit region*: If a philosopher is in the exit region, then at some later point he/she enters the thinking region.

The following C1 and C2 are important criteria to evaluate the performance of an algorithm for the group k-exclusion problem.

(C1) *Waiting time in the trying region*: The time from when a philosopher wishes to enter a forum until he/she enters the forum.
(C2) *k-concurrent entering*: If P_i requests a forum, and if the number of fora requested by all philosophers is at most k and remains so until P_i enters the forum, then P_i enters the forum within a finite length of time.

3. Vidyasankar's algorithm

The following procedure, (n, m, k)-*VidGME*, given by Vidyasankar [9], is an algorithm for the group k-exclusion problem.

```
procedure (n, m, k)-VidGME
shared variables
  for every i ∈ {1, ..., n}: forum(i) ∈ {0, 1, ..., m}, initially 0, writable by
    a process i and readable by all processes j ≠ i; level(i) ∈ {0, ..., n − 1},
    initially 0, writable by a process i and readable by all processes j ≠ i;
  for every s ∈ {1, ..., n − 1}: turn(s) ∈ {1, ..., n}, initially arbitrary,
    writable and readable by all processes;

process i
  input actions {inputs to process i from philosopher P_i}:
    try_i(f) for every 1 ⩽ f ⩽ m, exit_i;
  output actions {outputs from process i to philosopher P_i}: talk_i, think_i;

  ** thinking region **

01: try_i(f):
02:     forum(i) := f;
03:     for s := 1 to n − 1 do begin
04:       level(i) := s;
05:       turn(s) := i;
06:       waitfor |{forum(u): s ⩽ level(u), 1 ⩽ u ⩽ n}| ⩽ k
                or turn(s) ≠ i end;
07:     talk_i;

  ** talking region **

08: exit_i:
09:     level(i) := 0; forum(i) := 0; think_i;
```

As stated in [9], procedure (n, m, k)-*VidGME* satisfies group k-exclusion, k-lockout avoidance, progress for the exit region, and k-concurrent entering. However, Vidyasankar did not estimate waiting time for a process in the trying region. We estimate an upper bound on waiting time in the trying region of (n, m, k)-*VidGME* in the rest of this section.

In an execution by (n, m, k)-*VidGME*, process i is said to be a winner at s if it has left the **waitfor** statement at line 06 in the sth loop of the **for** statement at line 03. Note that if a process is a winner at level s then the process is also a winner at any level t $(1 \leqslant t \leqslant s)$. For each i $(1 \leqslant i \leqslant n)$, when process i has entered the exit region, the qualification for the winner of process i is cancelled by resetting $level(i) := 0$ at line 09. The proof of the next

lemma is essentially the same as the proof of the n-processes algorithm by Peterson [6] for the mutual exclusion problem, and we omit it here.

Lemma 1. *For any execution by (n, m, k)-VidGME, if the number of different fora requested by processes at level s or higher levels (i.e., sth loop or higher loops of the* **for** *statement at line* 03) *is greater than k at a point in time, then there are at most $n - s$ winners at level s at the point in time.*

Theorem 1. *Suppose that all processes are faultless. In any execution by (n, m, k)-VidGME, waiting time for a process in the trying region is bounded by $\mathrm{O}(n^2)l$ if $m \leqslant k$ or $n \leqslant k$, and by $\frac{n(n-k)}{k}c + \mathrm{O}(\frac{n^3(n-k)}{k})l$ if $k < m$ and $k < n$.*

Proof. Let $F(s)$ be a time bound from when a particular process enters level s (i.e., the sth loop of the **for** statement at line 03) until it becomes a winner at level s.

If $m \leqslant k$ or $n \leqslant k$, the first condition of the **waitfor** statement at line 06 is always satisfied in the execution by (n, m, k)-*VidGME*. Hence, in this case, $F(s) = \mathrm{O}(n)l$ for $1 \leqslant s \leqslant n - 1$. Then the time for a particular process to stay in the trying region is bounded by $\mathrm{O}(n^2)l$.

We next consider the case where $k < m$, $k < n$ and $n - k < s \leqslant n - 1$. If the number of different fora requested by processes at level s or higher levels is more than k, then the number of different fora at level $s - 1$ or higher is also more than k. From Lemma 1, the number of processes at level s or higher levels is at most $n - (s - 1) > k$. This is contrary to $n - k < s$. Hence, for a process at level s or a higher level, the first condition of the **waitfor** statement at line 06 is always satisfied, so $F(s) = \mathrm{O}(n)l$.

We finally consider the case where $k < m$, $k < n$ and $1 \leqslant s \leqslant n - k$. Let process i be the last arrival at level s at a point τ in time and not a winner at level s at τ. The most time consuming scenario for process i at level s to move to level $s + 1$ is the case where there are $n - s > k$ winners at level s, who are not yet winners at level $s + 1$, at τ and these winners request as many as different fora so that the first condition of the **waitfor** statement is not satisfied for level s. Let us consider such a scenario for process i. Then the winners at level s can move to level $s + 1$ within $\mathrm{O}(n)l$ time from τ. If the number of winners at a level is not more than k, these winners can move to the next level by the first condition of the **waitfor** statement. Since a process is pushed up by a later arrival, at least k processes can move smoothly to level $n - k + 1$, level by level, by the second condition of the **waitfor** statement. (Note that we consider the worst case where there are more than k winners at each level lower than level $n - k + 1$.) From Lemma 1, the number of different fora requested by processes at level $n - k + 1$ or higher levels is at most k. It follows that any process at level $n - k + 1$ can gain smooth access to the entrance of the talking region, level by level, since the first condition of the **waitfor** statement is satisfied. Hence, for each level s ($1 \leqslant s \leqslant n - k$), the first k winners at level $s - 1$ can move smoothly level by level by the first condition or the second condition of the **waitfor** statement. Other processes at level s will be pushed up by a new arrival (by the second condition of the **waitfor** statement), or they move to level $s + 1$ when the number of winners at level $s - 1$ becomes not more than k (by the first condition of the **waitfor** statement). Therefore, we have the following equality:

$$F(s) = \frac{(n-s)}{k}\big(c + \mathrm{O}\big((n-s+1)n\big)l\big).$$

Hence, if $k < m$ and $k < n$, the time from when a particular process enters the trying region until it enters the talking region is bounded by

$$\sum_{s=1}^{n-1} F(s) = \sum_{s=1}^{n-k} F(s) + \sum_{n-k+1}^{n-1} F(s) = \frac{n(n-k)}{k}c + \mathrm{O}\left(\frac{n^3(n-k)}{k}\right)l. \qquad \square$$

The proof of Theorem 2 is similar to that of Theorem 1, and we omit it here.

Theorem 2. *Suppose that the number of stopping failures of processes is at most $k-1$. In any execution by (n,m,k)-VidGME, waiting time for a process in the trying region is bounded by $O(n^2)l$ if $m \leqslant k$ or $n \leqslant k$, and by $n(n-k)c + O(n^3(n-k))l$ if $k < m$ and $k < n$.*

The time bounds given in Theorems 1 and 2 are tight in the sense that there are possible scenarios for a process to reach these bounds.

4. Speedup of the algorithm

By Lemma 1, we may reduce the number of loops of the **for** statement at line 03 of (n,m,k)-*VidGME* to $n-k$ from $n-1$. However, speedup by this reduction is marginal unless $n-k = o(n)$. By incorporating a technique used in k-exclusion algorithm in [2,5] into (n,m,k)-*VidGME*, we can substantially reduce an upper bound on the waiting time in the trying region. In (n,m,k)-*SUGME* given below, we reduce the number of iterations in the **for** statement at line 03 from $n-1$ to $n-k$, and add one condition, $|\{u\colon\ level(u) \geqslant s,\ 1 \leqslant u \leqslant n\}| \leqslant n-s$, in the **waitfor** statement at line 06. This additional condition plays an important role in reducing the waiting time.

procedure (n,m,k)-*SUGME*
shared variables
  for every $i \in \{1,\ldots,n\}$: $forum(i) \in \{0,1,\ldots,m\}$, initially 0, writable by
    a process i and readable by all processes $j \neq i$; $level(i) \in \{0,\ldots,n-k\}$,
    initially 0, writable by a process i and readable by all processes $j \neq i$;
  for every $s \in \{1,\ldots,n-k\}$: $turn(s) \in \{1,\ldots,n\}$, initially arbitrary,
    writable and readable by all processes;

process i
  input/output actions: the same as the input/output actions of
      (n,m,k)-*VidGME*;

  ** thinking region **

01: $try_i(f)$:
02: $forum(i) := f$;
03: **for** $s := 1$ **to** $n-k$ **do begin**
04: $level(i) := s$;
05: $turn(s) := i$;
06: **waitfor** $|\{forum(u)\colon\ s \leqslant level(u),\ 1 \leqslant u \leqslant n\}| \leqslant k$
        or $|\{u\colon\ level(u) \geqslant s,\ 1 \leqslant u \leqslant n\}| \leqslant n-s$ **or** $turn(s) \neq i$ **end**;
07: $talk_i$;

  ** talking region **

08: $exit_i$:
09: $level(i) := 0$; $forum(i) := 0$; $think_i$;

In an execution by (n,m,k)-*SUGME*, winners at level s can be defined in the same way as in an execution by (n,m,k)-*VidGME*.

Lemma 2. *For any execution by (n, m, k)-SUGME, if the number of different fora requested by processes at level s or higher levels is greater at level s than k at a point in time, then there are at most $n - s$ winners at level s at the point in time.*

Proof. The proof is an induction on levels. For the sake of the contrary, we assume that there are n winners at level 1 at some point in time τ and the number of different fora requested by processes is greater than k at τ. Suppose that $turn(1) = i$ at τ. Then $turn(1)$ was set most recently by process i before τ. For any $j \neq i$, process j set $level(j)$ to be 1 before the time when $turn(1)$ was set by process i. Hence, for each $j \neq i$, $level(j) \neq 0$ during the time period from when process i set $turn(1)$ to be i until τ. Then all conditions of the **waitfor** statement for process i are not satisfied during the time period. Hence, process i cannot be a winner at level 1 at τ. This is contrary to the assumption that all the processes are winners at level 1 at τ.

Let $s \geqslant 2$. Assume that the assertion of the lemma holds for any $r < s$. For the sake of the contrary, we assume that there are more than $n - s$ winners at level s at a point in time τ, and the number of different fora requested by processes at level s or higher levels at τ is greater than k. Then for any $1 \leqslant r < s$, the number of different fora requested by processes at level r or higher levels is greater than k at τ. From the inductive hypothesis, there are at most $n - r$ winners at level r at τ. Among the winners at level s at τ, let process i be the process that set $turn(s)$ most recently before τ. Then for any process j $(j \neq i)$, among the winners at level s at τ, $level(j) \geqslant s$ during the time period from when process i set $turn(s)$ to be i until τ. Hence, during this period, the second condition of the **waitfor** statement for process i at level s is not satisfied. Since we assumed that the first condition of the **waitfor** statement is not satisfied at τ and that process i is a winner at level s at τ, the third condition of the **waitfor** statement for process i should be satisfied at some point in time during the time period. This means that some process, say process p, not in the set of the winners at level s at τ, must set $turn(s)$ to be p at some point in time during the time period. Then process p is a winner at level $s - 1$ at that point in time. Then the number of winners at level $s - 1$ at that point in time is more than $n - s + 1$ and the number of different fora at τ is greater than k. This is contrary to the inductive hypothesis. □

The existence of any number of process failures of the stopping type does not affect the proof of Lemma 2. Hence, from Lemma 2 we have the next theorem.

Theorem 3. *(n, m, k)-SUGME guarantees group k-exclusion even if any number of process failures of the stopping type exist.*

Progress for the exit region is obvious. In order to prove k-lockout avoidance, it is sufficient to give an upper bound on the waiting time for a faultless process in the case where at most $k - 1$ stopping failures exist.

Theorem 4. *Suppose that the number of process failures of the stopping type is at most $k - 1$. In any execution by (n, m, k)-SUGME, waiting time for a faultless process in the trying region is bounded by $O(n(n - k))l$ if $m \leqslant k$ or $n \leqslant k$, and by $(n - k)c + O(n(n - k)^2)l$ if $k < m$ and $k < n$.*

Proof. For each s, $1 \leqslant s \leqslant n - k$, let $F(s)$ be an upper bound on the time from when a faultless process has entered level s until it becomes a winner at level s. The worst situation is the case where $\min\{n, k\} - 1$ faulty processes are stopping at the entrances or the insides of $k - 1$ different meeting rooms. If $m \leqslant k$ or $n \leqslant k$, then $F(s) = O(n)l$ and the waiting time for a faultless process in the trying region is bounded by $O(n(n - k))l$.

We now consider the case where $k < m$ and $k < n$. The worst situation for a faultless process i at level $n - k$ to reach the talking region is the case where the number of different fora requested by processes at level $n - k$ including $k - 1$ faulty processes stopping at the entrances or the insides of different $k - 1$ meeting rooms is $k + 1$ (by Lemma 2), and process i is not yet a winner at level $n - k$. When process i recognizes that one process leaves a

meeting room, or when it is pushed up by a new arrival from level $n-k-1$, process i is allowed to enter a meeting room. Hence, $F(n-k) = c + \mathrm{O}(n)l$.

For a process i at level $1 \leqslant s \leqslant n-k-1$, if the first condition of the **waitfor** statement is satisfied, it can move to level s in $\mathrm{O}(n)l$ time. Suppose that the first condition is not satisfied at a point in time τ. From Lemma 2 the number of winners at level $s-1$ and level s are at most $n-s+1$ and $n-s$, respectively. Hence, by the second condition of the **waitfor** statement, $F(s) = c + \mathrm{O}((n-k-s)n)l$, and the waiting time for a faultless process in the trying region is bounded by

$$\sum_{s=1}^{n-k} F(s) \leqslant \sum_{s=1}^{n-k} \bigl(c + \mathrm{O}\bigl((n-k-s)n\bigr)l\bigr) \leqslant (n-k)c + \mathrm{O}\bigl(n(n-k)^2\bigr)l. \qquad \square$$

The major difference between (n,m,k)-*VidGME* and (n,m,k)-*SUGME* is that the second condition of the **waitfor** statement is added in the latter procedure. Since there are at most $n-s$ winners at level s (by Lemma 2), the last arrival at level s can move to the next level (by the second condition of the **waitfor** statement of (n,m,k)-*SUGME*) when at least one winner at level s exits the talking region, even if no new arrivals reach to level s before the point in time and other winners are very slow or stopping. Therefore, for (n,m,k)-*SUGME*, there is no substantial difference between the upper bounds of the running time in the trying region in the case where no faulty processes exist and in the case where there are at most $k-1$ faulty processes. However, there is a significant difference between these two cases for (n,m,k)-*VidGME* as shown by Theorems 1 and 2.

References

[1] M.J. Fischer, N.A. Lynch, J.E. Burns, A. Borodi, Resource allocation with immunity to limited process failure, in: Proc. 20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1979, pp. 234–254.
[2] Y. Igarashi, Y. Nishitani, Speedup of the n-process mutual exclusion algorithm, Parallel Process. Lett. 9 (1999) 475–485.
[3] Y.-J. Joung, Asynchronous group mutual exclusion, Distributed Comput. 13 (2000) 189–206.
[4] N.A. Lynch, Distributed Algorithms, Morgan Kaufmann, San Francisco, CA, 1996.
[5] M. Omori, K. Obokata, K. Motegi, Y. Igarashi, Analysis of some lockout avoidance algorithms for the k-exclusion problem, Interdiscip. Inform. Sci. 8 (2002) 187–192.
[6] G.L. Peterson, Myths about the mutual exclusion problem, Inform. Process. Lett. 12 (1981) 115–116.
[7] M. Takamura, Y. Igarashi, Group mutual exclusion algorithms based on ticket orders, in: 9th Annual International Computing and Combinatorics Conference (COCOON 2003), Big Sky, Montana, USA, July 25–28, 2003, in: Lecture Notes in Comput. Sci., vol. 2697, Springer-Verlag, Berlin, 2003, pp. 232–241.
[8] K. Vidyasankar, A highly concurrent group mutual l-exclusion algorithm, in: Proceedings of the 12th Annual ACM Symposium on Principles of Distributed Computing, Monterey, CA, 2002, p. 130.
[9] K. Vidyasankar, A simple group mutual l-exclusion algorithm, Inform. Process. Lett. 85 (2003) 79–85.