A New Self-Stabilizing k-out-of-ℓ Exclusion Algorithm on Rings

Ajoy K. Datta[1], Rachid Hadid[2], and Vincent Villain[2]

[1] School of Computer Science, University of Nevada Las Vegas
datta@cs.unlv.edu
[2] LaRIA, Université de Picardie Jules Verne
5, rue de Moulin Neuf, 80000 Amiens France
{hadid,villain}@laria.u-picardie.fr

Abstract. We present an efficient self-stabilizing solution to the k-out-of-ℓ exclusion problem on a ring. The k-out-of-ℓ exclusion problem is a generalization of the well-known mutual exclusion problem — there are ℓ units of a shared resource, any process can request at most k ($1 \leq k \leq \ell$) units of the shared resource, and no resource unit can be allocated to more than one process at one time. This solution is based on the circulation of ℓ tokens around the ring. A processor requesting NEED (NEED $\leq k \leq \ell$) units of the resource can enter the critical section only upon receipt of NEED tokens. We propose a simple and pessimistic method to handle the deadlock problem. So, after stabilization, no mechanism is needed for the deadlock detection. Moreover, in this paper, we give a formal definition of a new efficiency property, called (k, ℓ)-*liveness*, which is a desirable property of any k-out-of-ℓ exclusion solution. This property allows as many processors as possible to execute their critical sections simultaneously without violating the safety property. We generalize the technique introduced in [6] to maintain the right number (ℓ) tokens in the system. The tokens are counted without using any counter variable for all processors except one, called the Root. This solution improves the waiting time of an earlier solution [4] by maintaining a reasonable stabilization time. The waiting time is reduced from $(\ell + 2)(n - 1)$ to $2(n - 1)$, where n is the size of the ring. The stabilization time is $8n$ instead of $4n$ in [4]. One nice characteristic of our algorithm is that its space requirement is independent of ℓ for all processors except the Root.

Keywords: Fault-tolerance, k-out-of-ℓ exclusion, mutual exclusion, ℓ-exclusion, self-stabilization.

1 Introduction

The ℓ-*exclusion problem* is a generalization of the mutual exclusion problem — ℓ processors are allowed to execute the critical section concurrently. This problem models the situation where there is a pool of ℓ units of a shared resource and each processor can request at most one unit. The k-out-of-ℓ *exclusion* problem allows every processor to request at most k ($1 \leq k \leq \ell$) units of the shared

S.-T. Huang and T. Herman (Eds.): SSS 2003, LNCS 2704, pp. 113–128, 2003.

resources, but no unit can be assigned to more than one processor at the same time [10] . One example of this type of resource sharing is the sharing of channel bandwidth. The bandwidth requirements vary among the requests multiplexing the channel. For example, the demand would be quite different for a video from an audio transmission request. Algorithms (not self-stabilizing) for k-out-of-ℓ exclusion were given in [3, 8, 7, 10, 11]. All these algorithms are permission-based — a processor can access the resource after receiving a permission from all the processors of the system [10, 11] or from the processors constituting the quorum it belongs to [8, 7]. The only self-stabilizing solution to the k-out-of-ℓ exclusion algorithm is presented in [4]. This solution is based on the circulation of ℓ tokens around the ring. A processor requesting k_i ($k_i \leq k \leq \ell$) units of the resource can enter the critical section only upon receipt of k_i tokens. It is shown in [4] that this simple circulation of ℓ tokens is prone to deadlocks. Intuitively, let α be the number of the (critical section entry) requesting processors in the system and β the total number of tokens requested by α processors. If $\beta \geq \ell + \alpha$, then ℓ tokens can be allocated in such a manner that every requesting processor is waiting for at least one token. So, the system has reached a deadlock configuration. The deadlock problem is solved in [4] by using additional control messages.

Contributions. In this paper, we present a self-stabilizing solution to the k-out-of-ℓ exclusion problem on rings. We propose a new solution to the deadlock problem raised in [4]. Contrary to [4], when the system is stabilized, we need no extra control messages for the deadlock detection. The main idea of our solution is that only one of the requesting processors is given the permission to collect enough tokens to enter the critical section. That is, the process of receiving the permission and getting ready to enter the critical section is implemented as a mutually exclusive task. We show that our solution satisfies the (k, ℓ)-liveness property. In addition to its simplicity, this algorithm improves the waiting time of algorithm in [4] from $(\ell + 2)(n - 1)$ to $2(n - 1)$. The system may start in an arbitrary, possibly a deadlocked configuration. Therefore, in the stabilizing solution, we use a deadlock detection and resolution scheme which is used only during the stabilization phase. This detection/resolution scheme is very simple and efficient, and has only a constant overhead both in terms of space and size of messages. We add only a two-bit field in the message. We use a very simple token controller scheme to count and maintain the right number (ℓ) of tokens in the ring. The controller does not generate any additional messages (as produced in [4]). The space requirement of our algorithm is independent of ℓ for all processors except the Root. The stabilization time of the protocol is $8n$.

Outline of the Paper. In Section 2, we describe the model used in this paper. We present the specification of the k-out-of-ℓ exclusion problem in Section 3. We first present a non-self-stabilizing k-out-of-ℓ exclusion protocol in Section 4. Then in Section 5, we extend that solution to design a self-stabilizing k-out-of-ℓ exclusion algorithm. We only present the proof outline of this self-stabilizing solution due to lack of space. Finally, we make some concluding remarks in