Gerard Tel, Friedemann Mattern:
Comments on "Ring Based Termination Detection Algorithm for Distributed Computations".
Information Processing Letters 31. pp. 127-128, 1989

Information Processing Letters 31 (1989) 127–128
North-Holland

8 May 1989

COMMENTS ON "RING BASED TERMINATION DETECTION ALGORITHM FOR DISTRIBUTED COMPUTATIONS"

G. TEL

Department of Computer Science, University of Utrecht, P.O. Box 80.089, NL 3508 TB Utrecht, The Netherlands

F. MATTERN

Department of Computer Science, University of Kaiserslautern, P.O. Box 3049, D-6750 Kaiserslautern, Fed. Rep. Germany

Communicated by R. Wilhelm
Received 28 November 1988

Keywords: Distributed termination, false termination detection, distributed program, distributed algorithm, message-based algorithm, decentralized control

In a recent paper, Haldar and Subramanian [4] have presented a distributed termination detection algorithm. We like to inform you that the algorithm is not correct: it can lead some processes to conclude termination in situations in which there is not (a so-called "false termination"). Indeed, the correctness argument (*Proof of assertion* (2), page 153) is incomplete and contains a serious error, which is commonly made in reasoning about distributed systems. Therefore, we elaborate on this proof here. First it is remarked that the return of an unfalsified detection message to its originator implies that it passed all processes in a passive state. Subsequently, it is erroneously concluded that all processes are passive. But a view of a distributed system obtained in this way is close to meaningless, because the processes were observed at different time instants. The observation that all processes were passive at some moment does not imply that at some moment all processes were passive. The proof mentions the control sections of processes, but fails to indicate how these prevent the algorithm from detecting a false termination.

For a concrete counterexample to the algorithm, consider a network with processes A and B (see Fig. 1). All processes other than A and B remain passive throughout the example, hence they have empty control sections all the time and forward every detection message (DM) in our counterexample. Initially, only process A is active, and the following sequence of steps takes place.

(1) A initiates a basic communication and thus activates B (now B is in the control section of A and $Farthest(A) = B$).
(2) A becomes passive and sends $DM(A, B, 0)$ to $succ(A)$.
(3) B reactives A (now A is in the control section of B and $Farthest(B) = A$).
(4) B becomes passive and sends $DM(B, A, 0)$ to $succ(B)$.

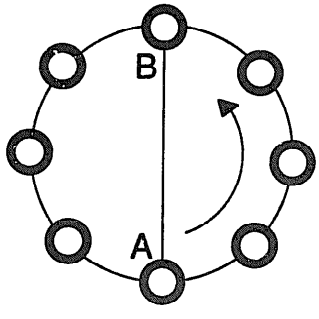

Fig. 1.

(5) B receives A's $DM(A, B, 0)$ (from step (2)) and executes lines 6–9 of page 152. After removing A from $ID(B)$, $ID(B)$ is empty and, because B is passive, B forwards $DM(A, B, 0)$.
(6) A reactivates B (now B is in the control section of A and $Farthest(A) = B$).
(7) A becomes passive and sends $DM(A, B, 0)$ to $succ(A)$.
(8) A receives B's $DM(B, A, 0)$ (from step (4)) and forwards it.
(9) A receives $DM(A, B, 0)$, as forwarded by B in step 5, and enters the termination phase, while B is active.

The delays that a detection message suffers on its way from A to B and vice versa can be arbitrarily long if a sufficient number of processes between A and B is assumed. In any case the delays can be so long that the above scenario is feasible. The counterexample relies on the possibility of reactivation of a passive process by a message from an active process. Without this possibility, which is usually assumed [2,3,6,7,8,9,10], the problem becomes trivial and is solved with an algorithm far simpler than proposed in [4].

We further remark the algorithm is not repaired by simply interchanging (on page 152) line 6 with lines 7–9 (and so with corresponding lines in other parts of the algorithm). After such a modification, the counterexample is modified as follows. Steps (1) to (4) are repeated, so that two DMs are underway from A to B. The first one is purged, but clears the control section, so that the second one is forwarded. Again A enters the termination phase while B is active. In general we feel that there is no way to "repair" a faulty algorithm. A counterexample suffices to show that an algorithm is incorrect, but this does not imply that avoiding this execution [1] yields a correct algorithm.

Unfortunately, more incorrect termination detection algorithms have been published in *Information Processing Letters* in the past [2,5], see also [9,10]. These examples clearly show the importance of a thorough, mathematically tight correctness proof for even the simplest distributed algorithm. Examples of such proofs, based on invariants, are found in [3,6].

We would like to point out that there exist a large number of distributed termination detection algorithms, including symmetric ones [8], with differing merits. An inventory and some new algorithms were given in [7,8]. The publication of over sixty papers on the subject in the past few years (see [7]) should make authors, referees, and editors sceptical. Before considering the publication of a new algorithm, it must be ascertained that the new paper contributes to the knowledge of the scientific community. Not only must the algorithm be correct and clearly presented, it should also be an improvement over existing solutions to the problem. In our opinion, the algorithm of [4], even if it were correct, is not.

References

[1] R.K. Arora and M.N. Gupta, More comments on "Distributed termination detection algorithm for distributed computations", *Inform. Process. Lett.* **29** (1988) 53–55.
[2] R.K. Arora, S.P. Rana and M.N. Gupta, Distributed termination detection algorithm for distributed computations, *Inform. Process. Lett.* **22** (1986) 311–314.
[3] E.W. Dijkstra, W.H.J. Feijen and A.J.M. van Gasteren, Derivation of a termination detection algorithm for distributed computations. *Inform. Process. Lett.* **16** (1983) 217–219.
[4] S. Haldar and D.K. Subramanian, Ring based termination detection algorithm for distributed computations, *Inform. Process. Lett.* **29** (1988) 149–153.
[5] C. Hazari and H. Zedan, A distributed algorithm for distributed termination, *Inform. Process. Lett.* **24** (1987) 293–297.
[6] F. Mattern, Global quiescence detection based on credit distribution and recovery, *Inform. Process. Lett.* **30** (1989) 195–200.
[7] F. Mattern, Algorithms for distributed termination detection, *Distributed Comput.* **2** (1987) 161–175.
[8] R.B. Tan and J. van Leeuwen, *General Symmetric Distributed Termination Detection*, Tech. Rept. RUU-CS-86-2, University of Utrecht, Utrecht, 1986.
[9] R.B. Tan, G. Tel and J. van Leeuwen, Comments on "Distributed termination detection algorithm for distributed computations", *Inform. Process. Lett.* **23** (1986) 163.
[10] G. Tel and J. van Leeuwen, Comments on "A distributed algorithm for distributed termination, *Inform. Process. Lett.* **25** (1987) 349.