An Efficient Mutual Exclusion Protocol in a Mobile Computing Environment

Sung-Hoon Park
Dept. of Computer
Chungbuk National Univ., Chungbuk, Korea

Abstract

The mutual exclusion (MX) paradigm can be used as a building block in many practical problems such as group communication, atomic commitment and replicated data management where the exclusive use of an object might be useful. The problem has been widely studied in the research community since one reason for this wide interest is that many distributed protocols need a mutual exclusion protocol. However, despite its usefulness, to our knowledge there is no work that has been devoted to this problem in a mobile computing environment. In this paper, we describe a solution to the mutual exclusion problem from mobile computing systems. This solution is based on the token-based mutual exclusion algorithm.

Key-words: *Synchronous Distributed Systems, Mutual exclusion, Fault Tolerance, Mobile Computing System.*

1. INTRODUCTION

The wide use of small portable computers and the advances in wireless networking technologies have made mobile computing today a reality. There are different types of wireless media: cellar (analog and digital phones), wireless LAN, and unused portions of FM radio or satellite services. A mobile host can interact with the three different types of wireless networks at different point of time. Mobile systems are more often subject to environmental adversities which can cause loss of messages or data [8]. In particular, a mobile host can fail or disconnect from the rest of the network. Designing fault-tolerant distributed applications in such an environment is a complex endeavor.

In recent years, several paradigms have been identified to simplify the design of fault-tolerant distributed applications in a conventional static system. Mutual exclusion, simply MX, is among the most noticeable, particularly since it is closely related to accessing shared resource called the critical section (CS) [7], which (among other uses) provides an exclusive access basis for implementing the critical section.

The mutual exclusion problem [1] requires two properties, safety and liveness, from a given set of processes. The problem has been widely studied in the research community [2-6] since one reason for this wide interest is that many distributed protocols need an mutual exclusion protocol. However, despite its usefulness, to our knowledge there is no work that has been devoted to this problem in a mobile computing environment.

The aim of this paper is to propose a solution to the mutual exclusion problem in a specific mobile computing environment. This solution is based on the token-based mutual exclusion algorithm that is a classical one for distributed systems. The rest of this paper is organized as follows: in Section 2, a solution to the mutual exclusion problem in a conventional synchronous system is presented. Section 3 describes the mobile system model we use. A protocol to solve the mutual exclusion problem in a mobile computing system is presented in Section 4. We conclude in Section 5.

2. MUTUAL EXCLUSION IN A STATIC SYSTEM

2.1 Model and Definitions

We consider a synchronous distributed system composed of a finite set of process $\Pi = \{p_1, p_2,., p_n\}$ connected by a logical ring. Communication is by message passing, synchronous and reliable. A process fails by simply stopping the execution (*crashing*), and the failed process does not recover. A correct process is the one that does not crash. Synchrony means that there is a bound on communication delays or process relative speeds. Between any two processes there exist two unidirectional channels. Processes communicate by sending and receiving messages over these channels.

The mutual exclusion problem is specified as following two

* *Corresponding author. E-mail: spark@chungbuk.ac.kr*
Manuscript received Dec.11, 2006 ; accepted Dec. 25, 2006

properties. One is for *safety* and the other is for *liveness*. The *safety* requirement asserts that any two processes connected the system should not have permission to use the critical section simultaneously. The *liveness* requirement asserts that every request for critical section is eventually granted. A mutual exclusion protocol is a protocol that generates runs that satisfy the mutual exclusion specification.

2.2 Token-based Mutual Exclusion Algorithm

As a classic paper, the token-based mutual exclusion algorithm, which was published by M. Raynal, specifies the mutual exclusion problem for synchronous distributed systems with crash failures and gives an elegant algorithm for the system; this algorithm is called the token-based MX Algorithm [2]. The basic idea in the token-based MX algorithm is that the any process holding the token can use the critical section exclusively. The token-based MX algorithm is described as follows.

- A distributed system is connected by a logical ring. Each process has a unique ID that is known by its neighborhood processes.
- The CS is exclusively used by the process holding the token.
- The token is circulated on the logical ring. If a process wants to use the CS, then it just waits until receiving a token from its neighborhood. Only when it has received the token, it has a right to use the CS exclusively.
- When the process with the token finished its use of CS, it immediately passes the token to its neighborhood.
- If a process doesn't to use a CS when it received the token, it just pass the token to it neighborhood.
- There exists only one token and the token is continuously circulated upon the logical ring.
- By doing this, any process eventually receives the token and it can use the CS exclusively, which means that this algorithm satisfies both of the safety and the liveness properties.

3. MOBILE SYSTEM MODEL

A distributed mobile system consists of two set of entities: a large number of mobile hosts (*MH*) and a set of fixed hosts, some of which act as mobile support stations (MSS_s) or base stations. The non *MSS* fixed hosts can be viewed as MSS_s whose cells are never visited by any mobile host. All fixed hosts and all communication paths connect them from the static network. Each *MSS* is able to communicate directly with mobile hosts located within its cell via a wireless medium. A cell is the geographical area covered by a *MSS*. A *MH* can directly communicate with a *MSS* (and vice versa) only if the *MH* is physically located within the cell serviced by the *MSS*. At any given instant of time, a *MH* can belong to one and only one cell. In order to send message to another *MH* that is not in the same cell, the source *MH* must contact its local *MSS* which forwards the messages to the local *MSS* of the target *MH* over the wireless network. The receiving *MSS*, in its turn, forwards the messages over the wireless network to the target *MH*. When a *MH* moves from one cell to another, a *Handoff procedure* is executed by the MSS_s of the two cells. Message propagation delay on the wired network is arbitrary but finite and channels between a *MSS* and each of its local mobile hosts ensure FIFO delivery of messages.

4. MUTUAL EXCLUSION IN A MOBILE SYSTEM

In the following, we consider a broadcast group G = (G_MSS, G_MH) of communicating mobile hosts, where G_MH and G_MSS are respectively a set of m mobile hosts roaming in a geographical area (like a campus area) covered by a fixed set of n MSS_s. In so far, local mobile hosts of base station MSS_i, which currently residing in MSS_i cell, will refer to mobile hosts that belong to group G.

A mobile host can move from one cell to another. If its current base station fails, the connection between the mobile host and the rest of system is broken. To recover its connection, a mobile host must move into another cell covered by an operational or correct base station. So, unless it crashes, a mobile host can always reconnect to the network. A mobile host may fail or voluntarily disconnect from the system. When a mobile host fails, its volatile state is lost.

In this environment, the mutual exclusion problem is defined over the set G_MH of mobile hosts. When a mobile host h_k wants to use the *CS*, it sends the request message to a *MSS*. In this case, the mobile host eventually should get the permission from the *MSS* and use the *CS*. Due to the resources constraints of mobile hosts and the limited bandwidth of wireless links, the distributed algorithm to solve mutual exclusion is executed by the set of *MSS* on behalf of the set G_MH of mobile hosts. In a first phase, the *MH* which wants to use the CS has to request the permission from the *MSS* in the cell which it belonging to. The *MSS* receiving those requests from the subset of G_MH of mobile hosts roaming in their respective cells keeps them in its queue. A token is circulated through the logical ring which consists of the fixed *MSSs*. In the second phase, when each *MSS* receives the token from its neighborhood, it sends the token to a mobile host h_k to give permission for the CS.

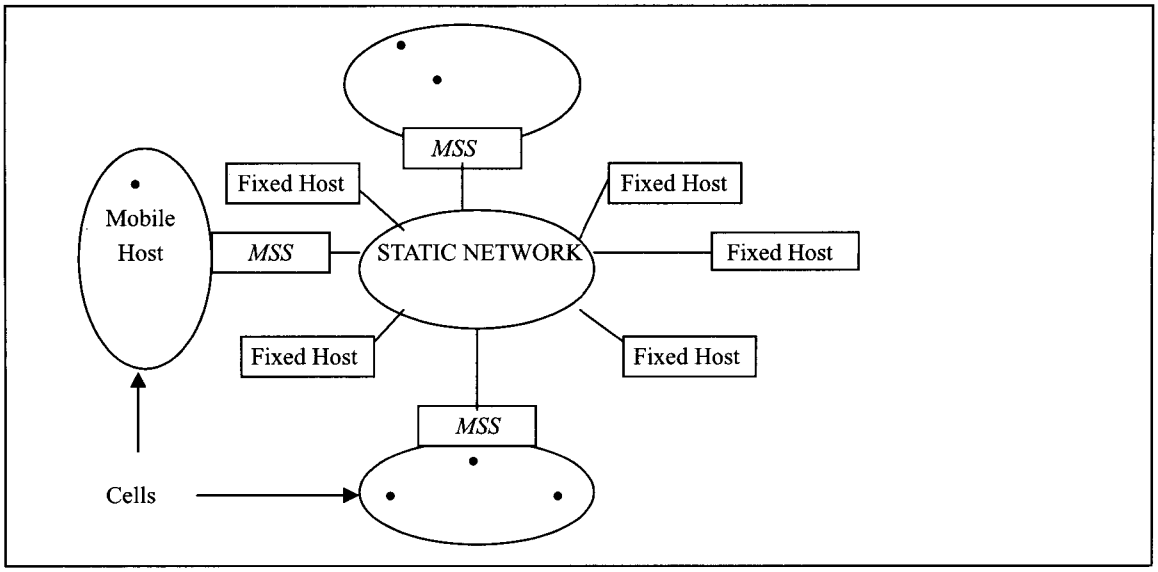

Fig. 1: Mobile System Model

Finally, the h_k received the permission from the *MSS* uses the CS and after using it returns the permission to the *MSS*. The *MSS* which has got the permission back from the h_k sends the token to the next turn of *MSSs*.

4.1 Principle

The mutual exclusion protocol proposed in this paper is based on the solution described by Raynal in Token-based MX algorithm [2]. The outlines of their protocol have been described in Section 2. In this section, we give an overview of our protocol and identify the major differences compared with the original token-based MX algorithm. We assume that the mutual exclusion is initiated by a mobile host which requests its current base station a token to use the CS. The contacted base station saves the request into the queue until it receives the token from its neighborhood.

During the mutual exclusion, each base station on one hand interacts with the mobile hosts located in its cell to gather the request of each mobile host for CS and on the other hand interacts with the other neighboring base stations to send and receive a token. In our approach, a base station *MSS* which participates in the mutual exclusion protocol, always acts on behalf of a subset of mobile hosts.

More precisely, the initial value of $Token\_\ Holdek_k$ is false but the value of it is changed true as a mobile host h_k that resides in MSS_i receives the token from its MSS_i. After returning the token to its base station, the mobile host h_k changes the value of its $Token\_Holder_k$ into false again.

The mutual exclusion protocol in such an environment consists of two cases depending on who the token holder is. As the first case, that is when a base station received a token from its neighboring base station or its mobile hosts. When it received the token from its neighboring base station, then it just sends the token to a mobile host with highest priority among the mobile hosts connected to the base station. In case of returning the token from its mobile hosts, it just sends the token to the next base station.

In this case the base station is doing two tasks concurrently, i.e., sending a token a mobile host with highest priority among the mobile hosts connected to the base station and receiving the token request message from mobile hosts concurrently. Thus the base stations play a role of a mutual exclusion coordinator during the mutual exclusion period.

During the mutual exclusion in a mobile computing environment, a base station playing a role of a mutual exclusion coordinator is needed to reduce the message traffic among mobile hosts. The mutual exclusion algorithm among base stations is similar to the token-based MX in static distributed systems. That is, only the base station holding the token has a permission to use the *CS*.

In the second case, that is when a mobile host received a token from its host base station. Then it just uses the *CS* for a while and returns the token to its host base station after finishing it.

In above scenario, we don't consider the mobility of the mobile host in the MX algorithm. But if we consider the mobility of the mobile host, then it makes the MX problem more complicated than the one of static distributed systems.

The differences of mutual exclusions between mobile computing environments and static distributed systems are as follows:

1) During the period of the MH using the CS, the MH changes its base station from the one that it received the token to the

other base station. In this case, the MH simply sends the token the base station of the cell in which it resides. But the base station that received the token takes some action to keep the fairness of the MX. The base station that did not send the token but received the token from its MH simply sends it to the base station which waits for the token to keep the fairness of the MX. Because, as a big difference between mobile computing environments and static distributed systems, the mobile host with token will appear in any cell whenever mutual exclusion protocol has started. Therefore, every base station should check all other base stations to know which base station cover the mobile host holding the token in the cell. That causes message traffics among base stations.

2) In mobile computing environment, a handoff algorithm is needed to perform mutual exclusions correctly, but it is not needed in static distributed systems.
3) Due to the resource constraints of mobile hosts and the limited bandwidth of wireless links, the distributed algorithm to solve mutual exclusion is executed by the set of MSS_s on behalf of the set *G_MH* of mobile hosts.

4.2 Protocol

The protocol is composed of three parts and each part contains a defined set of actions. Part A (fig. 2) describes the role of an arbitrary mobile host h_k. Part B (fig. 3) presents the protocol executed by a base station MSS_i.

```
% Mobile host h_k is located in MSS_i cell %
(1) Upon receipt of the request for CS from the
    application
      Send Req_Token to MSS_i
(2) Upon receipt of Token from MSS_i
% The mobile host (h_k) gets into CS %
      CS (h_k)
(3) Upon receipt of the release for CS from the
    application
      Send Release_Token to MSS_i
```

Fig. 2. Protocol Executed by a Mobile Host h_k (Part A)

Part B is related to the interactions between a base station and its local mobile hosts on one hand and the other base station on the other hand. Thus, Part B is based on the traditional Token-based MX protocol adapted to our environment.

Finally, the part C of the protocol is the handoff protocol destined to handle mobility of hosts between different cells.

In fig. 2, the three actions performed by an arbitrary mobile host are:

(1) A mobile host executes this action when it receives a request from an upper application program to initiate a mutual exclusion.
(2) Token message is sent to a mobile host h_k by the mobile support systems MSS_i when it had requested a token from the local base station where it resides. Upon receipt of such a message, the mobile host gets into the *Critical Section*.
(3) When the application program terminates the mutual exclusion protocol, the Token is released to the mobile support system, MSS_i.

Actions of the protocol in fig. 3 numbered from (4) to (7) are executed a mobile support system, i.e., a base station MSS_i. They have the following meaning:

(4) When a base station is asked by a mobile host to send a Token, it inserts the request into the rear of its queue.

```
My_Staus_i := 0;
My_Queue_i := ∅;
Cobegin
(4) || Upon receipt of Req_Token( h_k )
       insert Req_Token(h_k) to rear (My_Queue_i);
(5) || Upon receipt of Token (MSS_i-1)
       if My_Queue_i ≠ ∅ then
         My_Status_i := 1;
          send Token to front (My_Queue_i);
          delete front (My_Queue_i);
       else
          send Token to MSS_i+1;
       end-if
(6) || Upon receipt of Token ( h_k )
    if ( Phase_i = 0 ∧ My_Queue_i ≠ ∅ ) then
         My_Status_i:= 1;
          send Token to front (My_Queue_i);
          delete front (My_Queue_i);
    else
         My_Status_i:= 0;
         send Token to MSS_i+1;
    end-if
(7) || Upon receipt of Req_Token ( MSS_j )
       insert Req_Token(h_k) to Rear(My_Queue_i);
```

Fig. 3. Protocol Executed by a mobile support station MSS_i (Part B)

(5) In case of receiving a Token from other base station, the base station checks its queue *My_ Queue*$_i$ to confirm whether the queue is empty or not. If the queue is not empty, then the base station sends the Token to the mobile host that is positioned at the front of the queue. And it deletes the element from the queue and sets its status to true that means it holding Token, i.e., *My_Status*$_i$:= 1. But

if the queue is empty, then the base station just passes the Token to the next base station.

(6) When a base station receives a Token from a mobile host h_k, it checks its queue and status. If both ($Phase_i = 0 \wedge My\_Queue_i \neq \varnothing$) are true, which means that it does not hold the token and at the same time the queue is not empty, then the base station sends the Token to the mobile host that is the front element of the queue. And it deletes the element from the queue and sets its status to true. Otherwise it sends the Token to the next base station and sets its status to false.

(7) On receiving the Token request message from other mobile support system, the MSS_i insert the request message into its queue.

As shown in Fig. 4, the handoff protocol is described.

(8) When a mobile host h_k moves from MSS_j cell to MSS_i cell, the handoff protocol execution is triggered. Mobile host h_k has to identify itself to its base station by sending a message GUEST(h_k, MSS_j).

(9) Upon receiving this message, MSS_i learns that a new mobile host h_k, coming from MSS_j cell has entered in its cell. With BEGIN_ HANDOFF(h_k, MSS_i) message, MSS_i informs MSS_j that it removes h_k from the set of mobile hosts that reside in its cell.

(10) Upon receiving such a message, MSS_j checks its queue to confirm that the token request of h_k is in the queue. If it is in its queue, then it transfers the token request to MSS_i and deletes the token request from the queue.

Cobegin
% Role of h_k %
(8) || **Upon** entry in MSS_i cell
send Guest(h_k, MSS_j) to MSS_i
% Role of MSS_i
(9) || **Upon** receipt of GUEST(h_k, MSS_j)
Local_MH_i := Local_MH_i $\cup$ {h_k};
send BEGIN_HANDOFF(h_k, MSS_i) to MSS_j;
% Role of MSS_j
(10)||**Upon** receipt of BEGIN_HANFOFF(h_k, MSS_i)
$Local\_MH_j := Loacl\_MH_j - \{h_k\}$;
If (Req_Token(h_k) $\in$ My_Queue$_i$) **then**
send Req_Token(h_k) to MSS_i;
delete Req_Token(h_k) from My_Queue$_i$;
end-if

Fig. 4. Handoff Procedure (Part C)

4.3 Correctness Proof

As our protocol is based on the Token-based logical ring algorithm proposed by M. Raynal, some statements of lemmas and theorems that follow are similar to the ones encountered in [2].

Theorem 1 No two different processes can have permission to use the critical section simultaneously (safety property).

Proof (proof by contradiction). Let assume that there exist two mobile hosts to get a permission to use the critical section. A mobile host can use the *CS* only if it received a permission token from the *MSS* of the cell to which it belonging (action 2). In this case, the assumption means that there exist two *MSS*s holding the token or one *MSS* sends the token twice to two different mobile hosts each. The first case is false since there is only one token circulating under the logical ring. The second case is also false since the *MSS* holding the token sends it to mobile host h_k only once (action 5). So it is a contradiction. □*Theorem* 1

Theorem 2 Every request for the critical section is eventually granted (liveliness property).

Proof If a mobile host sends a message to request a token (action 1), at least one *MSS* eventually receives it and inserts it into the queue (action 4). After that, there are two cases. In first case, if the mobile host h_k sent the message does not move to other cell, then the message Req_Token eventually will be positioned at the front of the queue and the *MSS* received the message sends the token. Thus, the mobile host sent the message eventually receives the token and uses the *CS*. In a second case, when the mobile host h_k sent a message Req_Token moves from MSS_j cell to another MSS_i cell before receiving the token, then the handoff protocol execution is triggered (action 8-10). Mobile host h_k has to identify itself to its base station by sending a message GUEST(h_k, MSS_j). In this case, by (action 10) the request message will be transferred to the MSS of the cell to which the mobile host has moved. Consequently, the mobile host will receive the Token and use the CS when the *MSS* sends the Token. □*Theorem* 2

5. CONCLUSION

The communication over wireless links are limited to a few messages (in the best case, three messages: one to request a token and the others to get the token and release the token respectively) and the consumption of mobile hosts CPU time is low since the actual mutual exclusion is run by the base stations. The protocol is then more energy efficient. The protocol is also independent from the overall number of mobile hosts and all needed data structures are managed by the base stations. Therefore, the protocol is scalable and can not be affected by mobile host failures.

In addition, other interesting characteristics of the protocol are as follows. 1) During the mutual exclusion period, a base station should keep track of every mobile host within its cell to manage the request messages and the token. 2) In such a mobile computing environment, a handoff algorithm is needed to perform mutual exclusions efficiently and correctly, but it is not needed in static distributed systems.

The mutual exclusion algorithm in a mobile computing environment consists of two important phases. One is a local mutual exclusion phase in which a mobile host holds and uses the CS. The other phase is a global mutual exclusion phase in which each *MSS* takes part in the mutual exclusion by passing the token to another *MSS*.

REFERENCES

[1] D. Agrawal, A.E. Abbadi, An efficient and fault-tolerant solution for distributed mutual exclusion, ACM Trans. Computing Systems 9 (1) (1991) 1.20.

[2] M. Raynal, Algorithms for Mutual Exclusion, MIT Press, Cambridge, MA, 1986.

[3] M. Maekawa, A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems, ACM Trans. Computer Systems 3 (2) (1985) 145.159.

[4] D. Manivannan, M. Singhal, An efficient fault-tolerant mutual exclusion algorithm for distributed systems, in: Proceedings of the ISCA International Conference on Parallel and Distributed Computing Systems, 1994, pp. 525.530.

[5] K. Vidyasankar, A simple group mutual ℓ-exclusion algorithm, Inform. Processes. Letter 85 (2003) 79.85.

[6] M. Singhal, A taxonomy of distributed mutual exclusion, J. Parallel Distributed. Computing 18 (1) (1993) 94. 101.

[7] S. Lodha, A.D. Kshemkalyani, A fair distributed mutual exclusion algorithm, IEEE Trans. Parallel Distributed Systems 11 (6) (2000) 537.549.

[8] Pradhan D. K., Krichna P. and Vaidya N. H., Recoverable mobile environments: Design and tradeoff analysis. FTCS-26, June 1996.

[9] Alagar S., Venkatesan., Causally ordered massage delivery in mobile systems, in proc. Of Workshop on Mobile Computing Systems and Applications, Santacruz, CA, Dec. 1994.

[10] Badache N., Mobility in Distributed Systems, Technical Report #962, IRISA, Rennes, Octor 1995.

[11] Badrinath B.R, Acharya A. and Imielinski T., Impact of mobility on distributed computat-ions, ACM Operating Review, 27(2), April 1993.

Sung-Hoon Park

He received the B.S in economics and statistics from Korea university in 1982, M.S in Computer science from Indiana University USA in 1991 and received Ph.D. in computer science and engineering from Korea university in 2000. In 2004, he has been an associate professor in chungbuk national university Korea. His main research interests include distributed system, mobile computing and theory of computation.