이동 분산 시스템에서 상호배제 알고리즘의 설계

Design of a Mutual Exclusion Algorithm in Mobile Distributed Systems

박성훈
충북대학교 전기전자컴퓨터공학부
Sung-Hoon Park(spark@cbnu.ac.kr)

요약

상호배제 패러다임은 그룹통신, 원자적 커밋먼트 프로토콜이나 복제 데이터 관리 등의 하나의 객체에 대한 독점적인 이용이 필요한 실질적인 문제들의 해결방안을 위한 기본적인 수단으로 이용 될 수 있다. 이러한 문제는 그 동안 광범위하게 연구자들에게 연구되어 왔던 바, 그 이유는 많은 분산 프로토콜들이 상호배제 프로토콜을 필요로 하기 때문이다. 그러나 이러한 유용성에도 불구하고 아직까지 이동 컴퓨팅 환경에서 이러한 문제를 다룬 적은 별로 없었다. 본 논문에서는 이동 컴퓨팅 시스템 하에서 상호배제의 문제를 기술코자 한다. 본 논문에서 제시하는 해결방안은 토큰 기반의 알고리즘에 기반을 두게 된다.

■ 중심어 : □동기적인 분산시스템□상호배제 | 결함 허용 | 이동컴퓨팅 시스템 |

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.

■ keyword : □Synchronous Distributed Systems□Mutual Exclusion□Fault Tolerance□Mobile Computing System□

I. 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[1]. 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.

* 본 논문은 2005학년도 충북대학교 학술연구지원사업의 연구비 지원에 의하여 연구되었습니다.

접수번호 : #061208-001
접수일자 : 2006년 12월 08일
심사완료일 : 2006년 12월 21일
교신저자 : 박성훈, e-mail : spark@cbnu.ac.kr

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)[2], which (among other uses) provides an exclusive access basis for implementing the critical section.

The mutual exclusion problem[3] requires two properties, safety and liveness, from a given set of processes. The problem has been widely studied in the research community[4-8] 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. In Section 5, we address the practical observation. We conclude in Section 6.

II. Mutual Exclusion in a Static System

2.1 Model and Definitions

We consider a synchronous distributed system composed of a finite set of process $\Pi = \{p1, p2, .., pn\}$ 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 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[4]. 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.

III. Mobile System Model

3.1 The Structure of Mobile System

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*) 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.

3.2 Characteristics of Mobile Hosts

The bandwidth of the wireless link connecting a *MH* to a *MSS* is significantly lower than bandwidth of the links between static hosts [9]. In addition, mobile hosts have tight constraints on power consumption relative to desktop machines, since they usually operate on stand-alone energy sources such as battery cells. Consequently, they often operate in a doze mode or voluntarily disconnect from the network.

Transmission and reception of messages over wireless links also consume power at a *MH*. So, distributed algorithm for mobile systems need to minimize communication over wireless links. Furthermore, mobile hosts are less powerful than fixed hosts and have less memory and disk storage. Hence, while designing distributed algorithms for mobile systems, the following factors should be taken into account[10][11]:

- The amount of computation performed by a mobile host should be kept low
- The communication overhead in the wireless medium should be minimal.
- Algorithm should be scalable with respect to the number of mobile hosts.
- Algorithm should be able to easily handle the effect of mobile host disconnections and connections.

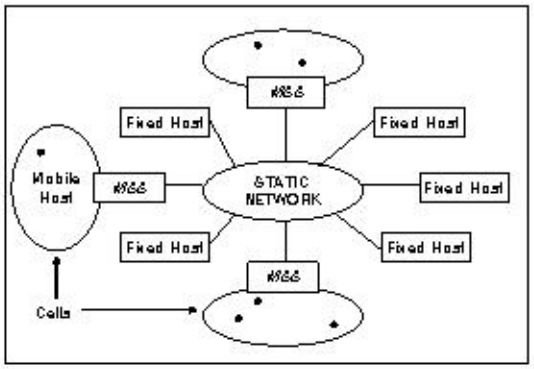

Fig. 1. Mobile System Model

IV. 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_x 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 MSS_s. In the second phase, when each MSS receives the token from its neighborhood, it sends the token to a mobile host h_x to give permission for the CS.

Finally, the hk 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 hk sends the token to the next turn of MSS_s.

4.1 Principle

The mutual exclusion protocol proposed in this paper is based on the solution described by Raynal in Token-based MX algorithm[4]. 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_Hold ek_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.

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 hk (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:

```
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 MSSi (Part B)

(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.

(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 hk, it checks its queue and status. If both ($Phase_i = 0 \ \wedge \ My\_Queue_i = \varnothing$) 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 hk moves from MSS_i cell to MSS_j 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_j learns that a new mobile host h_k coming from MSS_i cell has entered in its cell. With BEGIN_ HANDOFF(h_k, MSS_i) message, MSS_j informs MSS_i that it removes hk from the set of mobile hosts that reside in its cell.

(10) Upon receiving such a message, MSS_i 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_i) to MSS_i
    % Role of MSS_i
    (9) || Upon receipt of GUEST(h_k, MSS_i)
  Local_MH_i := Local_MH_i ∪ {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 := Local_MH_j {h_k}
      If (Req_Token(h_k) My_Queue_j ) then
        send Req_Token(h_k) to MSS_i
       delete Req_Token(h_k) from My_Queue_j
      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 [4].

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_i). 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*

V. A Practical Observation

We address here some practical observations from the very structure of our mobile MK algorithm on the

solvability of Mutual Exclusion. A corollary of our result above is that we can design mutual exclusion algorithm more efficiently based on the mobile support systems instead of the mobile hosts, and but that solves the mutual exclusion.

Is this only theoretically interesting? We believe not, as we will discuss below. 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.

Interestingly, the mobile host h_k moves around many cells regardless of its holding the token. If we consider the mobility of the mobile host, then it makes the MX problem more complicated than the one of static distributed systems.

As the differences of mutual exclusions between mobile computing environments and static distributed systems, our MX algorithm has the following merits:

First, during the period of the MH h_k using the CS, the MH h_k 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. In this MX algorithm, the base station that received the token should take some action to keep the fairness of the MX. That is, when 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. 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. Thus, 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 .In our MX algorithm, to prevent those message traffic caused by above stations, only two base stations together take part in keeping the mobility of the MH.

Second, in mobile computing environment, a handoff algorithm is needed to perform mutual exclusions correctly, but it is not needed in static distributed systems. If a MH moves from one cell to the other cell during the periods of using the CS, then the handoff algorithm is invoked to keep trace of the MH holding the token. Eventually both base stations know the location of the MH with token and decide how to manage the token.

Third, due to the resource constraints of mobile hosts and the limited bandwidth of wireless links, the designed distributed algorithm to solve mutual exclusion is executed by the set of *MSSs* on behalf of the set *G_MH* of mobile hosts.

VI. 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*

참 고 문 헌

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

[2] M. Singhal, "A taxonomy of distributed mutual exclusion," J. Parallel Distributed Computing, Vol.18, No.1, pp.94-101, 1993.

[3] D. Agrawal and A. E. Abbadi, "An efficient and fault-tolerant solution for distributed mutual exclusion," ACM Trans. Computing Systems, Vol.9, No.1, pp.1-20, 1991.

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

[5] M. Maekawa, "A √N algorithm for mutual exclusion in decentralized systems," ACM Trans. Computer Systems, Vol.3, No.2, pp.145-159, 1985.

[6] D. Manivannan and M. Singhal, "An efficient fault-tolerant mutual exclusion algorithm for distributed systems," Proceedings of the ISCA International Conference on Parallel and Distributed Computing Systems, pp.525-530, 1994.

[7] K. Vidyasankar, "A simple group mutual l-exclusion algorithm," Inform. Processes. Letter85, pp.79-85, 2003.

[8] S. Lodha and A. D. Kshemkalyani, "A fair distributed mutual exclusion algorithm," IEEE Trans. Parallel Distributed Systems, Vol.11, No.6, pp.537-549, 2000.

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

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

[11] B. R. Badrinath, A. Acharya, and T. Imielinski, "Impact of mobility on distributed computations," ACM Operating Review, Vol.27, No.2, Apr. 1993.

저 자 소 개

박 성 훈(Sung-Hoon Park) 종신회원

- 1982년 2월 : 고려대학교 정경대학 통계학과 (경제학사)
- 1993년 2월 : 인디애나대학교 대학원 컴퓨터학과(공학석사)
- 1997년 2월 : 고려대학교대 컴퓨터학과 (공학박사)
- 2004년 9월 ~ 현재 : 충북대학교 전기전자컴퓨터공학부 부교수

<관심분야> : 분산/모바일/유비쿼터스 시스템 및 알고리즘