이동 데이터베이스 시스템을 위한 효율적인 이타적 잠금기법

권 혁 신* · 김 세 윤* · 김 응 모**

An Efficient Altruistic Locking Protocol for the Mobile Transaction Management System

Hyeok Shin Kwon* · Se Yoon Kim* · Ung Mo Kim**

Abstract

We propose an advanced transaction scheduling protocol to improve the concurrency and to guarantee the mobility for the mobile database management systems. Mobility, portability, and wireless link in mobile computing environment can cause certain drawbacks, and thus it is more difficult to solve the concurrency control problems. However, a locking scheme should be used to guarantee the data consistency and to prevent the data conflicts. It is well known that data consistency is guaranteed by standard transaction scheduling schemes like *two-phase locking* (2PL). It has two of operation, lock and unlock. But 2PL does not give solution for mobile system. *Altruistic Locking* (AL) and classifying transactions, we adapt, can give solution for the previous problems. AL, as an advanced protocol, has attempted to reduce delay effects associated with lock release moment by the use of donation. In this paper, we extend those approaches and classify the transactions to reduce delay effects of short-lived transaction caused by long-lived transaction. In addition, we show efficient solution for the case of disconnection occurrence. Our protocol, namely, *Mobile Altruistic Locking* (MAL) is shown to be efficiently used in order to reduce delay effects and to guarantee database consistency in a state of the slippery connection in mobile database systems.

Keywords : Mobile Transaction, Locking Protocol, Altruistic Locking, Concurrency Control

---

논문접수일 : 2003년 10월 20일 논문게재확정일 : 2003년 12월 22일
※ 본 연구는 대학 IT 연구센터 육성지원사업의 연구결과로 수행되었음.
* 성균관대학교 정보통신공학부 컴퓨터공학과 석사과정
** 성균관대학교 정보통신공학부 컴퓨터공학과 교수

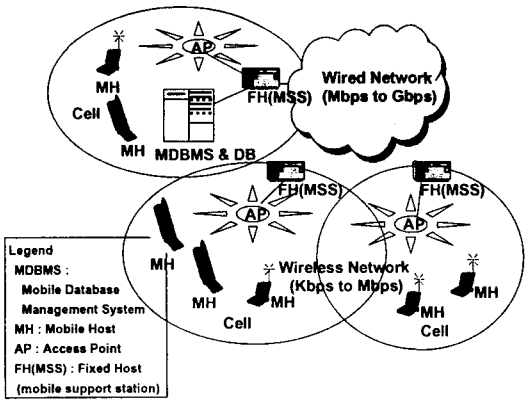
1. Introduction

Recent advances in information technology have provided portable computers with mobile interface that allow networked communication even while users are mobile. Low bandwidth, higher error rates, and more frequent disconnection characterize mobile communication [1]. So we must guarantee that executing transactions can quickly access databases with other conflicting transactions. We also must guarantee that they are always in a consistent state without losing correctness of locked data items, even in case of disconnection by handoff or low bandwidth. It is well known that data consistency is guaranteed by standard transaction scheduling schemes like 2PL(two-phase locking). AL(Altruistic Locking) is basically an extension of 2PL in the sense that several transactions hold locks on an object simultaneously under certain conditions. However, these locking protocol reveal inefficiency in mobile computing environment when disconnection occurs because of unilateral abort, unilateral commit by fixed host or factors of mobile environment. MAL(Mobile Altruistic Locking), our protocol, is based on AL. MAL has some new methods, namely, restart management, management of classified transactions, and expansion of AL for mobile transaction management are additionally used in order to improve the concurrency and satisfy serializability in mobile computing environment.

2. Related Works

2.1 Mobile System Architecture

Advances in computing and networking technologies have made extensive use of portable computers possible and enabled on-line information sharing via wireless communication channels. This new computing paradigm, called mobile computing, allows users to perform on-line transaction processing independent of their physical locations [4]. Generally, such a mobile computing architecture includes two distinct sets of entities basically : Mobile Hosts (MHs) in the wireless network and Mobile Support Stations (MSS) in the wired network <Figure 1>. The MSSs are steadily connected to the wired network and some of devices, called Access Points (APs), are augmented with a wireless interface to communicate with the MHs. The MHs can dynamically move within a radio coverage area called a cell or between two cells while retaining their network connections. Normally, a single AP is able to support a number of MHs, and is en-



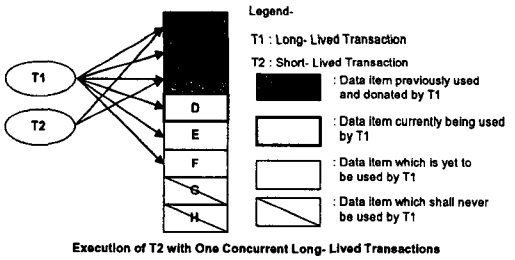
<Figure 1> Mobile System Architecture

gaged to provide services such as data passing and message interpretation to the MHs positioned only within its cell.

MSS is responsible for keeping track of addresses of MHs to detect the geographical location where the MH is located. After a mobile host requests to process a transaction to a fixed host, then the fixed host which is connected to database begins to execute the operations of the transaction. In replicated mobile database environments, multiple MHs maintain replicated data and they use replication control tools for data synchronization. In the concurrency control view, this situation makes overhead of system and is not good for efficient processing. There could be frequent disconnections and abort behavior caused by low bandwidth, slippery connections and frequent movement. That frequently causes rollback actions. If the disconnection occurs when the transaction connect to DBMS to execute some operations, it has to restart operations from first operation in traditional protocols like 2PL. That causes inefficiency of concurrency control. Our proposal solves these problems by using the altruistic locking scheme.

2.2 Altruistic Locking

AL is a modification of 2PL under certain conditions. AL provides a third concurrency control operation, called Donate, along with Lock and Unlock. Donate operation is managed to let scheduler know that the long-lived transaction, owner of the object, does not need to access that object which was no longer required [5]. Several rules control the use of the donate operation by data management system. Transactions can only donate objects that they currently have locked. However, they may not access any object that they have donated. A well-organized transactions must unlock every objects that it locked, regardless of whether it donated any of those objects. Transactions are never required to donate any objects ; donations are always optional. Donate operations are beneficial since they can permit other transactions to lock the donated object before it is unlocked. To favor any other short transactions, they could be allowed to access data that are still being grasped by a certain long transaction yet to be committed. In AL, this way of endowment is established by declaring a set of data to be consigned, wake, by long transactions.



<Figure 2> Altruistic Locking Protocol Architecture

See the <Figure 2>. T1 is LLT(long-lived transaction) and T2 is SLT(short-lived transaction). T1 locked object A, B, C, D, E and F. T1 previously used data item A, B, C and donated successfully. Now T1 is being phase of execution with D. T2 request A and C to lock. In 2PL, that is not allowed by scheduler and T2 has to wait in wait-queue until T1

release all objects locked. However, the request to A and C by T2 can be allowed in AL by using donate operation. In AL, T2 is not allowed to access object G or H and it can cause some of delay. In MAL, that is allowed under certain condition by using classifying the transactions. That could reduce delay effects. These contents are going to mention in section 3.

3. Proposed Protocol

3.1 Basic Concepts

In the mobile environment, the concurrency control is very important to guarantee consistency and efficient processing of transactions. To solve these problems, we took AL protocol that is more efficient to process transactions than 2PL. It can reduce delay effect by use donate operation. However, there are cascade roll-backs when the long-lived transaction is aborted and that can not give the guarantee necessity of commit in the mobile environment. More situation than these effects are not enough to appropriate to the mobile system, so we extended AL and classified the transactions to give more chances to lock objects early than AL. In the disconnections occur, we use system log file. This is good for restart transactions to continue at the end of the execution not at the first operation. The AL assumes that the long-lived transactions eventually commit. This assumption is not appropriate to mobile system, because disconnections could occur frequently in this environment. So, the AL has to be modified suitably. To prevent that transactions which have some of donated objects are aborted frequently, we use the concepts of strict 2PL and modified commit point of time. In addition, we lay emphasis on the recovery of aborted transactions. To make good efficient concurrency control protocol, we classified the transactions by using character of transaction.

3.2 Assumptions

To describe our protocol, Mobile Altruistic Locking (MAL), in detail, the following assumptions are made mainly with regard to transaction management principle in mobile environment.

(1) ***Deadlock Handling*** : If a transaction happens to fall into deadlock situation, the transaction will be eliminated by using a certain deadlock timeout scheme.
(2) ***Transaction Priority*** : A restarted transactions is vested higher priority than any other transaction, say, either short-transaction or long-transaction.
(3) ***Transaction Restart*** : All aborted transactions can be restarted only when user has the aborted transaction issue a request of re-execution.
(4) ***Marking Transaction*** : All transactions has own information about what operations will be executed and what objects will be used.

3.3 Locking Protocol

We applied AL(altruistic locking) protocol

because that can make transaction which used donated objects can commit earlier than donating transaction do without violating. However, we reorganized that to proper to mobile environment. We explain out proposed protocol in this section. What concepts are used and how to be processed transactions, we will discuss.

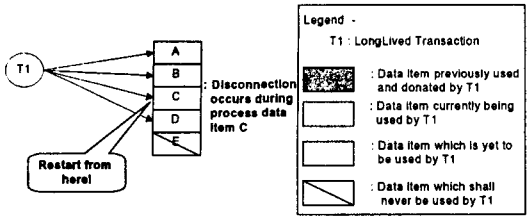
3.3.1 Reorganizations of AL and classify the transactions

We use marking transaction. The marking transaction descries it has own information of what data item will be locked and used. So, if the transaction occurs, first of all it locks all of objects needed and then executes operations and finally unlocks all locked objects. While the transaction execute operations with objects sequentially, if the transaction will not be more use some of previously used objects, that objects could be donated. And then, if some transaction use this donated objects, the information of this order remains on the log file. To guarantee serializability of transaction when the disconnection occurs, we reorganized rule.

1) Transaction which has donated objects is only allowed to lock request to some objects which are not in the donating transaction's wake when the transaction's request operation is read. Donating transaction means the transaction which already used some objects and donated. This is a method for recovery in mobile computing.
2) If the read-only transaction is using some objects and other transaction request to same objects with write operation, the system creates replica for requesting transaction to be allowed to access without delay. On the other hand, if the other transaction sends request with read operation, it is allowed because of public-lock. Public-lock means the lock which is made by read operations from some transactions. Our protocol, MAL, can be explained as follows.

(1) Management of disconnection

In the mobile environment, the reorganized policy to improve efficiency of transaction concurrency control is needed to solve the problem of inefficiency when transaction restarts from disconnections or other causes. Suppose that disconnection occurs while executing the operations. Most of conventional locking protocols, such as 2PL, have to take an abort action (e.g. roll-back) and then restart the processing from state of the first operation. However, in MAL protocol, that could restart operations from state of disconnecting under certain condition without roll-back operations. Instead, the MAL protocol stores contents of donated data items in a log file. Suppose disconnection was occurred while transaction was processing with objects C in <Figure 3>. System does not need do roll-back action immediately because the transaction can do its operation continuatively after restart. If other transaction locked and used the donated objects A and B while T1 disconnected and restart after, that is not problem if the transaction could restarts from state of disconnection continuatively and then finally commits successfully its operations.



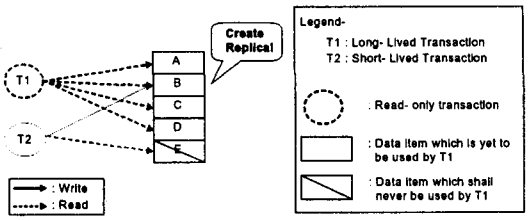
〈Figure 3〉 Processing of disconnected transaction

However, if other transactions locked and modified the objects C and D which were locked but were not executed until disconnection occurred by T1, the T1 could not obtain locks and could not restart continuatively. So, in this case, the transaction T1 has to roll-back and restart from first operation unavoidably and system informs that to other transactions which are using or used and committed A and B.

Finally, if the transaction is restarted without any violating, it can do continue processing with the objects at disconnection occurred.

(2) Classify transactions & create replica

We classified transactions to two. One is read-only transaction and the other is write transaction. The read-only transaction is a transaction which requests objects only with read operations. The other has read and one of write operations at least to objects. Assume the lock holder is read-only transaction and then lock requester transaction request to lock same object with write operation, this request can not be allowed until the lock holder unlock all of its objects in 2PL. However, in our proposal, MAL, that can be allowed by using create replica. The replica is given to lock holder transaction by system and original object is given to lock requester transaction. So, the lock request transaction can do his operation without waiting like T2 in <Figure 4>.



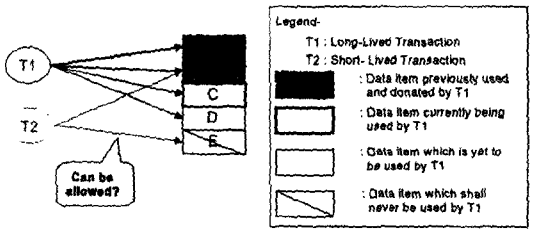
〈Figure 4〉 The create replica

(3) Expansion of access control

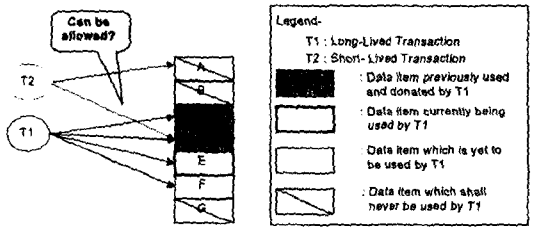
As MAL is operated on the mobile environment, all of transactions must have ability to do roll-back operations. In AL, the transaction must follow the wake. That is, if the transaction used donated object by others, it could use only donated object in the wake of donating transaction. This restriction can prevent cascade roll-back. However, we expanded access allowance to ensure efficiency on the situation of abort and roll-back with some restriction. If the transaction wants obtain lock to the object that is not in the donating transaction's wake, only read-only transaction can do. In the opposite direction, only read-only transactions that already locked some objects which is not using by any transaction can obtain the locks which are donated by other transaction.

In <Figure 5>, assume the T2 used B which were donated already by T1 with read or write operation. If the T2 want to continue processing with data item E with write operation, the request is not allowed. On the other hand, if T2 is read-only transaction, the request is

read properly and it can be allowed. That is, if the transaction which is using donated object wants to access other objects which are not in the wake, it is only allowed when the request transaction is read-only transaction. Assume T1 is aborted and it has to be roll-backed, data item A, B and C are going to be roll-backed. In this situation, data item E is not necessary to be roll-backed because T2 use that only read operation. So, although the abort action occurs in the donating transaction, it doesn't make problems in the situation that read-only transaction requests locks. Also, T2 can donate E. <Figure 6> is same as <Figure 5>.



<Figure 5> Restriction of Processing for recovery 1

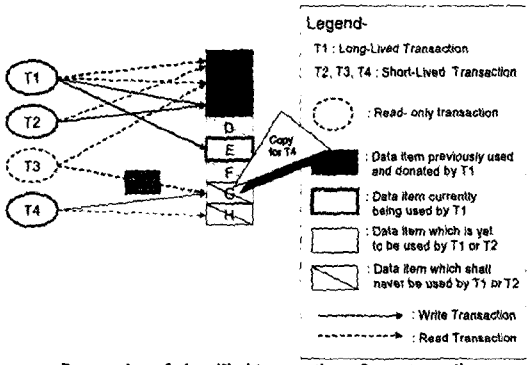


<Figure 6> Restriction of Processing for recovery 2

These restrictions can expand the scope of AL and these property can prevent cascade roll-back appropriately for recovery. We can get more efficiency to have chance to obtain lock than AL without cascading roll-back.

3.3.2 Operations of MAL

As the MAL is extension of AL, we classified transactions to two, read-only and write, to increase degree of concurrency control and recovery in mobile database system. We also use replica under the certain condition and guarantee efficiency of restart transaction. Our proposal reduces delay of transaction efficiently. Let's see the <Figure 7>.



<Figure 7> Proposed Protocol (MAL) Architecture

Suppose that a long transaction, say, T1 attempts to access data items A, B, C and E in an orderly manner. Note that data items D, F, G and H shall never be accessed by T1. Presume that T1 has already locked and thereafter donated A, B and C. Now T1 is supposed in the stage of accessing E. In the case of 2PL, transaction T2, T3 and T4 should have delay. However, in MAL, T2 could fortunately be allowed to access A and C without any delay because that objects are donated objects by T1. T3 also could be allowed to access B and G without any delay because object B is a donated by T1 and T3 is read-only transaction. T4 could be allowed to access

G and H without any delay because object G was locked with read-only by T3 and that time T4 want to use that with write operation, so system create replica for T4 and give that to T3. Finally, all of the transactions in figure 7 can do their operations simultaneously without delay.

Suppose the disconnection occurs when T1 is in processing of E. If the transaction want to restart and data item E are not used by other transaction (situation that lock is read and request is read on E is not problem as a matter of course) between disconnecting time and restart time, all of transactions do not cause problems just like disconnection was not occurred. However, if T1 has to roll-back, object G and H is not targets of abort action in MAL. Only system informs abortion to T2 and T3 without roll-back action of them.

In the view of mobile system architecture, mobile host which has already requested processing of T1 tries to re-connection. After re-connection, a scheduler in fixed host examines the requested data items, A, B, C, E and the donated data items, A, B, C which have been stored in log file. Data manager check where other transactions worked with object E between the times of disconnection occurs and restart or not. If E was not used by other transactions, it can do restart. Donated data items, A, B and C do not affect final value executed by T1 at all, if the transaction can restart. In other words, in this case, mobile host that has requested T1 is always guaranteed to get correct values of A, B and C without roll-back action. This is due to that both final results of T1 solely without any other intervening transactions and final results of T1 which donated objects A, B and C and restart from E with intervening transactions simultaneously produces always the same value. Thus, T1 can proceed from E continuously without having to restart from A.

3.4 Scheme of MAL Protocol

MAL algorithms can be pseudo-coded as follows.

```
Algorithm (MAL)

Input : SRT, LLT
/* SRT : Short lived transaction, LLT : Long lived
transaction */

/*** The part of management of restart ***/
BEGIN
 IF (SLT is restart transaction) THEN
   ReplyExam = ScheduleExam(SLT);
   /* Check SLT could violate serializability or not
     from using log-file */
  IF (ReplyExam = 0) /* SLT will not violate
                        serializability */
   SLT.Pri = HP ; /* Give high priority to SLT */
   ScheduleRestart(SLT) ; /* In this phase,
                             scheduler locate SLT
                             front line of queue */
  ELSE
   SendRes(SLT) ; /* Order SLT to restart from first)*/
   Return( ) ;
  ENDIF
 ENDIF
END

/******* The part of lock management *******/
BEGIN
 FOREACH SLT /*Whenever transaction is generated*/
  FOREACH SLT.TargetObject
  /* Number from number of object which
     transaction wanted */
    IF (SLT already has some objects) THEN /*1*/
```

```
IF (SLT does not use yet objects was used by
    any LLT ) THEN                          /* 2 */
 IF (Object.Lock = 0) THEN /* 3, The object was
                          not locked by some */
   Reply : = ScheduleLock(SLT)
 ELSE /* The object was locked by LLT */
   IF (Object.Donated = 1) THEN /* 4, The
     object was donated by some LLT */
     IF (SLT is read-only transaction) THEN
                                        /* 5 */
       Reply : = ScheduleLock(SLT)
     ELSE
       Reply : = ScheduleWait(SLT)
     ENDIF /* End of 5 */
   ELSE /* The object was not donated */
     IF (Object.Operation = 0) THEN /* 6, The
       object was locked by read */
       IF (LLT is read-only) THEN /* 7 */
         IF (SLT want read lock) THEN /* 8 */
           Reply : = ScheduleLock(SLT)
         ELSE
           Reply : = ScheduleCreateReplica(SLT)
         ENDIF /* End of 8 */
       ELSE
         IF (SLT is read-only transaction) THEN
                                        /* 9 */
           Reply : = ScheduleLock(SLT)
         ELSE
           Reply : = ScheduleWait(SLT)
         ENDIF /* End of 9 */
       ENDIF /* End of 7 */
     ENDIF /* End of 6 */
   ENDIF /* End of 4 */
 ENDIF /* End of 3 */
ELSE /* Transaction have used already donated
     object by some LLT */
 IF (Object.Lock = 0) THEN /*10, The object was
    not locked by any LLT */
   IF (SLT is read-only transaction) THEN /* 11 */
     Reply : = ScheduleLock(SLT)
   ELSE
     Reply : = ScheduleWait(SLT)
   ENDIF /* End of 11 */
 ELSE
   IF (Same ID) THEN
   /* 12, Donating transaction's ID that lock request
     transaction has and transaction's ID that have
     locked already the object */
     IF (Object.Donated = 1) THEN /*13, The object
                 was donated by the LLT */
       Reply : = ScheduleLock(SLT)
   ELSE
     IF (Object.Operation = 0) THEN /*14 */
       IF (LLT is read only transaction) THEN /* 15*/
         IF ( SLT wants read lock) THEN /* 16 */
           Reply : = ScheduleLock(SLT)
         ELSE
           Reply : = ScheduleCreateReplica(SLT)
         ENDIF /* End of 16 */
       ELSE
         IF (SLT is read-only transaction) THEN /*17*/
           Reply : = ScheduleLock(SLT)
         ELSE
           Reply : = ScheduleWait(SLT)
         ENDIF /* End of 17 */
       ENDIF /* End of 15*/
     ELSE
       Reply : = ScheduleWait(SLT)
     ENDIF /*End of 14 */
   ENDIF /* End of 13 */
     ELSE /* Different ID */
       IF (Object.Donated = 1) THEN /* 18 */
         IF (SLT is read-only transaction) THEN /* 19 */
           Reply : = ScheduleLock(SLT)
         ELSE
           Reply : = ScheduleWait(SLT)
         ENDIF /* End of 19*/
       ELSE
         IF (LLT is read-only transaction) THEN /*20*/
           IF (SLT want read lock) THEN /* 21*/
             Reply : = ScheduleLock(SLT)
           ELSE
             Reply : = ScheduleCreateReplica(SLT)
           ENDIF /* End of 21 */
         ELSE
           IF (Object.Operation = 0) THEN /* 22 */
             IF (SLT is read-only transaction) THEN
                                        /* 23 */
               Reply : = ScheduleLock(SLT)
             ELSE
               Reply : = ScheduleWait(SLT)
             ENDIF /* End of 23*/
           ELSE
             Reply : = ScheduleWait(SLT)
           ENDIF /* End of 22 */
         ENDIF /* End of 20 */
       ENDIF /* End of 18 */
     ENDIF /* End of 12 */
   ENDIF /* End of 10 */
 ENDIF /* End of 2 */
ELSE
```

```
    IF (Object.Lock = 0) THEN /* 24 */
      IF (Object.Donated = 1) THEN /* 25 */
        Reply : = ScheduleLock(SLT)
      ELSE
        IF (Object.Operation = 0) THEN /* 26 */
          IF (SLT wants read lock) THEN /* 27 */
            Reply : = ScheduleLock(SLT)
          ELSE
            Reply : = ScheduleCreateReplica(SLT)
          ENDIF /* End of 27*/
        ELSE
          Reply : = ScheduleWait(SLT)
        ENDIF /* End of 26 */
      ENDIF /* End of 25 */
    ENDIF /* End of 24 */
  ENDIF /* End of 1 */
ENDFOR
  IF (Reply = Abort) THEN /* Lock request of SLT is
                             aborted */
    Abort_Transaction(TransactionID) ;
    Send(Abort) ;
    Return() ;
  ENDIF
  ENDFOR
END
```

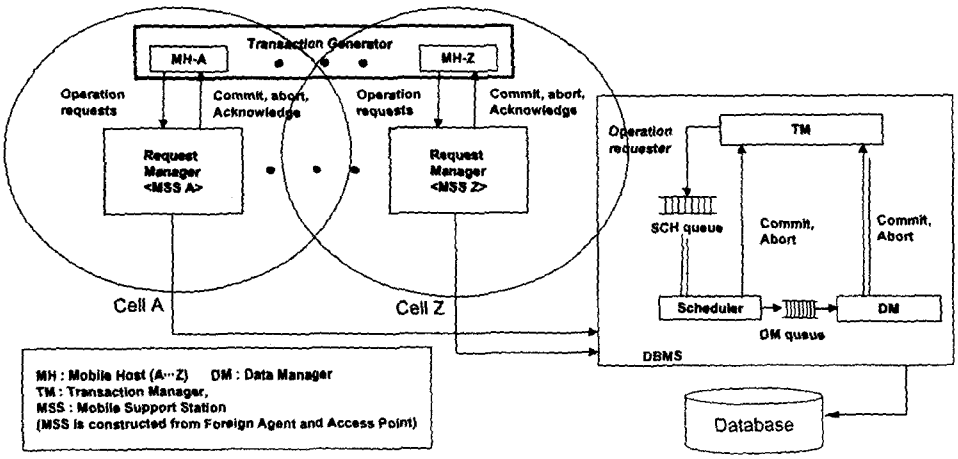
4. Performance Evaluation

In this chapter, we describe experimented performance behavior of MAL. Performance comparison is made against 2PL under various workloads. Major metrics chosen are transaction throughput and average transaction waiting time.

4.1 Simulation Model

4.1.1 Queuing System Model

The simulation model <Figure 8> consists of sub-components in charge of fate of a transaction from time of inception to time of retreat at each cell : Transaction Generator (TG), Transaction Manager(TM), Scheduler (SCH), Data Manager(DM), Database(DB). We have MSSs and each MSS has its own cell and access point. TG consists of a number of terminals. When TG generates transactions using exponential distribution, each transaction is marked with mobile host number and contents what objects will be used and what operations will be used. TG generates transactions one after another and sends their



<Figure 8> Simulation Model

operations to TM in a way of interleaving. TM receives transactions from terminals and passes them into SCH queue and then TM could receive acknowledge messages informing abortion from SCH or completion of a requested operation from DM. SCH could distinguish restart transaction and give chance to request object earlier than any other transactions using log files. DM analyzes an operation from SCH to determine which data item the operation is intended to access with what operation, and then sends the operation to the disk where the requested data item is stored. Also, DM check the transaction wants to restart and give objects to that transaction using log files. Whenever an operation is completed at the server, it sends to TM the message informing that the requested operation has been completed successfully.

This simulation model has been implemented using super-Simlib. Simlib is a set of FORTRAN subroutines for simulation study. The simlib routines appear in Law and Kelton : Simulation Modeling and Analysis. Super-Simlib is ANSI-C version of Simulation library routines [6]. In Super-Simlib formalism basic models must be specified from larger ones, and described how these ones are connected together in hierarchical fashion [7].

4.1.2 Experimental Methodology

<Table 1> summarizes the model parameters and shows the range of parameter values used in our experiments. Values for parameters were chosen by reflecting real world computing practices.

Number_mobile_hosts means the number of mobile hosts. We use two of transaction generator and they generate transactions using inter arrival time based on exponential distri-

〈Table 1〉 Parameters Setting for Simulation

Parameters	Values	Contents
Mumber_mobile_hosts	Over 2	The number of mobile hosts
Database_size	100	The number of object to be accessed
Transation_size (Short_lived_transaction_size, Long_lived_transaction_size)	Short : 1~5 Long : 6~20	The number of object which each transaction can request
Transaction_timeout	15~30	The time size which the available period of transaction (millisecond)
Inter_arr_time	5	Inter arrival time
Freq_arr_time	20~70%	Frequency of read_only transaction
Simulation_timeout	500~1500	Simulation running time (millisecond)
Ave_wait_time	Result value	The average time that transaction wait in queue
Freq_gen_repica	Result value	Frequency of generated replica (The value from result of simulation)
Throughput	Result value	Successful transactions against whole occurred transactions

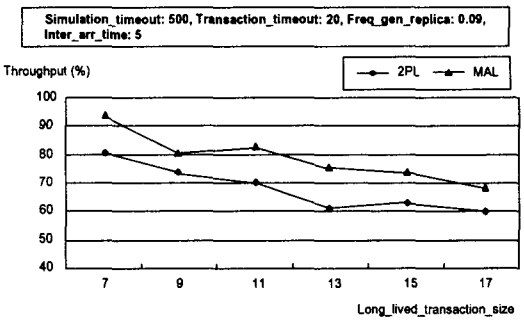
bution. Basically, inter arrival time is set to 5, so they work like several mobile hosts send transactions simultaneously. Database_size is set 100. Database size matters if it affects the degree of conflict. If *Database_size* is much larger than *Short_lived_transaction_size* and *long_lived_transaction_size*, conflicts rarely occur. When we set 100, the test models 2PL and MAL, shows different results obviously. Short_lived_transacion_size and Long_lived_transaction_size mean the number of objects which can be accessed by each transaction. That is, short lived transaction can request from 1 to 5 objects to lock. Long lived transaction can works with from 6 to 15 objects. Note that we define the 'Transaction_timeout' as a mean time of all transactions' life-time tested for in the paper. Transaction_size is random value within restricted numbers. Basically, we tested this protocol after set Simulation_length from 100 to 400. But results are not separated obviously for evaluation because test time is so short period. So, we set simulation time 500 basically.

4.2 Simulation Results and Interpretations

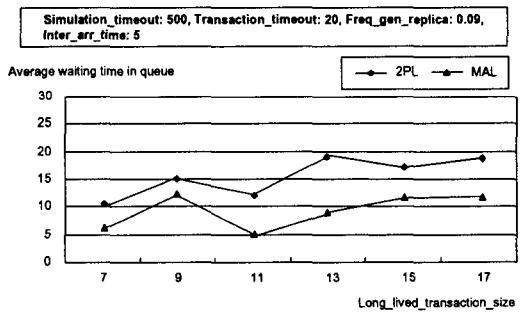
4.2.1 Effect of Multiprogramming Level

This experiment shows that MAL generally appears to outperform 2PL in terms of average waiting time. The good throughput performance is also exhibited by MAL.

Performance gain of *MAL* against *2PL* is from 8 to 14 percent increments in terms of throughput in most cases. And *MAL* outperforms *2PL* from 18 to 45 percent decrease of performance at transaction waiting time at long transaction size. This is because *MAL* has the 2PL plus the donation of data items, expansion of valid access scope and create replica when the read-only transactions exist. (<Figure 9> and <Figure 10>)



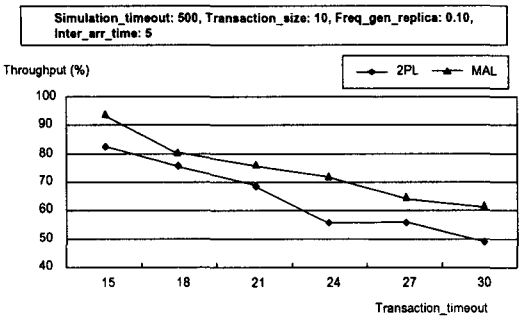
<Figure 9> Throughput with Size of Long Lived Transaction



<Figure 10> Average Waiting Time with Size of Long Lived Transaction

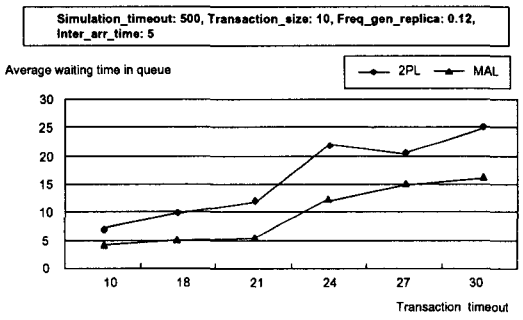
4.2.2 Effect of Timeout

At a higher range of transaction timeout, MAL shows higher throughputs <Figure 11> and a lower transaction waiting time <Figure 12>. As MAL's results, this phenomenon again shows us higher throughput gives lower average waiting time. MAL performs better than 2PL between 3 percent and 24 percent of performance at transaction throughput in most cases.



〈Figure 11〉 Throughput with Transaction Timeout

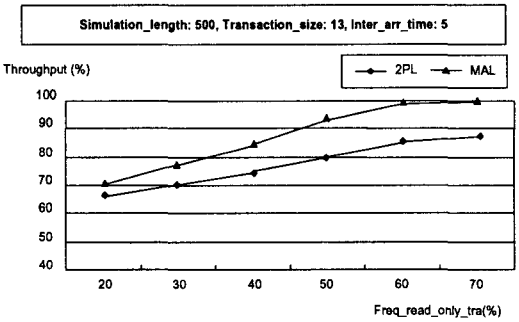
We can observe that average waiting time curve of 2PL more rapidly increase than MAL in <Figure 12>. As the transaction timeout is increased, the transaction waiting time of MAL is slowly increased. However, if the transaction timeout is far extended beyond a certain point, say 24, the average waiting time curve of 2PL more increase than MAL. If transaction timeout is larger than 30, the gap between 2PL and MAL will be wide.



〈Figure 12〉 Average Waiting Time with Transaction Timeout

4.2.3 Effect of Frequency of Read-only Transactions

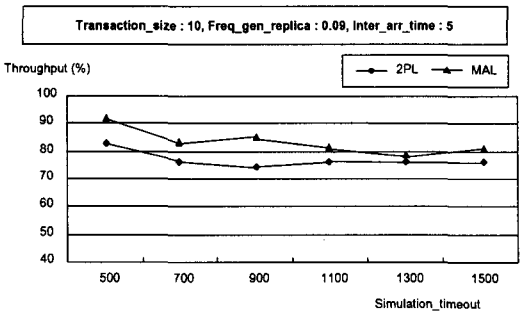
Most of cases, if more read operation exist, more efficient. In MAL outperform 2PL in this case <Figure 13>. We can get more efficient concurrency control in the environment which read-only transactions are frequent. When the read-only transaction exist 68%, MAL's transactions can works 100% of their job within simulation_length.



〈Figure 13〉 Average Throughput with Frequency of Read-only Transactions

4.2.3 Effect of Simulation Time

When we set simulation time to over 500, the difference of 2PL's and MAL's performance are shown obviously. MAL showed a more satisfying performance compared to 2PL about 10.2 percent on an average. Figure 14 is tested with 22% of read-only transactions. This results show a possibility that performance gain of 2PL against MAL could be deteriorated sharply if the simulation timeout size is far extended beyond a certain point.



〈Figure 14〉 Average Throughput with Simulation Time

Overall behaviors have been revealed that as the size of simulation timeout increases, MAL

generally outperforms in terms of throughput and waiting time.

5. Conclusion

We studied how to get more efficient concurrency for transaction control in mobile database system. We take three advantages in MAL, our proposed protocol, against 2PL. First of all is early release of lock using AL. The next is expansion of AL's scope and creation of replica using classified transactions. If there are more and more read-only request, transactions are more rapidly processed, because MAL release restriction to read-only transactions and replica without delay of transactions. Finally, restart mechanism. MAL showed a more satisfying performance compared to 2PL about 10.2 percent better. Conventional 2PL for mobile system is generally simple and convenient to implement in an environment that the short-lived transactions mainly access the database resources. However, as access needs for database in mobile environment are adapted to a wide range of applications, transaction processing models require long-lived transactions needs. In this case, MAL is recommended to improve the degree of concurrency where long-lived transactions naturally coexist with short-lived transactions and there are many read-only transactions in mobile database systems. We need more tests about MAL, an instance, efficiency with frequency of disconnections. In the future, we will study this protocol to apply to real-world in more detail for mobile database system.

References

[1] G.H. Forman, J. Zahorjan, "The Challenges of Mobile Computing", *IEEE Computer*, Apr., 1994

[2] P.A. Bernstein, V. Hadzilacos and N. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, Massachusetts, U.S.A., 1987.

[3] Kyong-I Ku and Yoo-Sung Kim, "Moflex Transaction Model for Mobile Heterogeneous Multi-database Systems", *Proceedings of IEEE 10th International Workshop on Research Issues in Data Engineering*, 2000.

[4] Siwoo Byun, Songchun Moon, "Resilient data management for replicated mobile database systems", *Data & Knowledge Engineering*, 29, 1999, pp. 43-55.

[5] K. Salem, H. Garcia-Molina and J., Shands, "Altruistic Locking", *ACM Transactions on Database Systems*, Vol. 19, No. 1, March, 1994, pp. 117-169.

[6] Law and Kelton, *Simulation Modeling and Analysis SUPER SIMLIB-C* Gregory, Glockner, School of Industrial and Systems Engineering Georgia Institute of Technology, August, 1993.

[7] N. Barbhouti and G. Kaiser, "Concurrency Control in Advanced Database Applications", *ACM Computing Surveys*, Vol. 22, No. 3, Sep., 1991, pp. 269-317.

[8] J. Lee and S. Son, "Performance of Concurrency Control Algorithms for Real-Time Database Systems, Performance of Concurrency Control Mechanisms in Centralized Database Systems", Prentice Hall,

1996, pp. 429-460.

[9] P.A. Bernstein and N. Goodman, "Multiversion Concurrency Control-Theory and Algorithms", *ACM Transactions on Database Systems*, December, 1983.

[10] J. Gray and A. Reuter, *Transaction Processing : Concepts and Techniques*, Morgan Kaufmann Publishers, Inc., 1993.

■ 저자소개

권 혁 신

1995년~2001년 시립인천대학교 전자계산 공학과 학사

2002년~2004년 성균관대학원 전기전자및컴퓨터 공학과 석사

김 세 윤

1994년~2001년 성균관대학교 전기전자및컴퓨터 공학과 학사

2002년~2004년 성균관대학원 전기전자및컴퓨터 공학과 석사

김 응 모

1977년~1981년 성균관대학교 수학과, 학사

1983년~1986년 Old Dominion University, 전산학과 석사

1986년~1990년 Northwestern University, 전산학과 박사

현재 성균관대학교 교수

◈ 이 논문은 2003년 10월 20일 접수하여 1차 수정을 거쳐 2003년 12월 22일 게재확정되었습니다.