論文2003-40CI-6-3

모바일 계산환경에서 거래 관리를 위한 동시성 제어

(Concurrency Control for Transaction Management in Mobile Computing)

李 惠 卿 *

(Hae-kyung Rhee)

요 약

데이터베이스 시스템에서 기존의 동시성 제어전략은 통신 중단이 빈번히 발생하는 모바일 컴퓨팅 환경하에서는 부적절하다. 모바일 컴퓨팅에서는 거래들을 처리하기 위한 시간이 극히 제한적이다. 이러한 시간을 가장 효율적으로 활용하기 위해서는 단기거래위주의처리가 바람직하다. 통신 중단으로 인한 Service Handoff는 특히 단기거래인 경우 모바일 거래들의 성능을 저하시킬 수 있다. 이타적 잠금 기법은 lock/unlock연산 이외에 donate 연산을 사용하기 때문에 단기거래 들에 대해 효율적이다. 모바일 컴퓨팅에서 이타적 잠금 기법을 사용했을 경우 단기거래의 성능은 향상될 수 있다.

Abstract

Traditional concurrency control strategies for database system are inappropriate for mobile computing environments in which communication disconnection could occur frequently. The time available to access transactions may be extremely limited in mobile computing. In order to ensure that this time is utilized most effectively, the short-lived transactions must be a main stream of transactions. Service handoffs which occur due to communication disconnection, could degrade the performance of mobile transactions especially for short-lived transactions. Altruistic locking protocol is in its nature favorable to short-lived transactions since it applies the donation of locks to long-lived transactions as early as possible. Performance for short-lived transactions could be improved when we applied altruistic locking to mobile computing.

Keywords : Mobile computing, Service handoff, Altruistic locking, Donate operation, Multidatabase

I. Introduction

As technological developmentsare made in software and hardware, users have become much more demanding in that they desire and require accessing information in anywhere, and at any time. Large amount of information that is accessible to a user at any given time is also growing at a rapid rate. Rapidly expanding technology is making available a wide breadth of devices with different memory, storage, network, power, and display requirements to access this diverse data set.

Traditional multidatabase systems(MDBS) are designed to allow timely and reliable access to large amounts of different data from different data sources in a distributed environment. Multidatabase resear-

* 正會員, 龍仁松潭大學 컴퓨터情報科 助敎授,
(Dept. of Computer Information at Yong-in Songdam college, Assistant professor)
接受日字:2003年5月27日, 수정완료일:2003年10月20日

chers have addressed topics such as autonomy, heterogeneity, transparency, and query resolution[1-5]. All of these solutions were based upon fixed clients and servers architecture under a reliable network infrastructure.

However, the concept of mobility, where a user accesses data through a remote connection with a portable device, has introduced additional difficulties and restrictions in a conventional database system. Access to shared data in a mobile environment is complicated by some fundamental constraints. These include: 1) a reduced capacity network connection, 2) processing and resource restrictions, and 3) effectively locating and accessing information from a multitude of sources. An MDBS with such additional restrictions is called a mobile data access system (MDAS).

A key requirement of this new world of computing will be the ability to access shared data regardless of location. Data from shared database systems must be made available to programs running on mobile computers. However, mobility poses serious impediments to meeting this requirement.

Traditionally, in a distributed environment, transactions are interleaved and executed concurrently[6] to achieve higher performance and throughput. The concurrent execution of transactions in an MDAS environment seems to be a more difficult job to control than in distributed database systems. This is due to the inferences among global transactions, inferences among global and local transactions, local autonomy of each sites, and frequently happened service handoffs. In order to reduce the effects of weak communications, data duplication should be used at the mobile unit. It provides additional availability in case of a communication failure.

In this paper, we apply a new concurrency control algorithm such as Altruistic Locking(AL)[8] to reduce the required communication overhead and get better performance of transactions.

Ⅱ. Background

Accessing a large amount of data over a limited capability network connection involves two general aspects: 1) the mobile networking environment and 2) mobility issues. The mobile environment includes the physical network architecture and access devices. Mobility issues include adaptability to a mobile environment, autonomy, and heterogeneity.

The concept of mobility implies a much more diverse range and amount of accessible data. A remote access system must provide access to a larger set of heterogeneous data sources. Multidatabase systems share many of the same characteristics as mobile systems.

1. Multidatabase Environment

A multidatabase system(MDBS) provides a logical integrated view and method to access multiple local database system while hiding the hardware and software intricacies from user. Access to the local DBMS through a much more diverse and restrictive communication and access devices is the natural extension to a traditional MDAS environment.

The main differentiating feature between an MDBS and an MDAS is the connection of server and/ or clients through a wirelessenvironment and the devices used to access the data. However, both environments are intended to provide timely and reliable access to the globally shared data. A summary of the issues facing a multidatabase and mobile system is given in Table 1[10]. We could find the similarities in the objectives of effectively accessing data in a multidatabase and mobile computing environment.

Due to the similarities in the objectives of effectively accessing data in a multidatabase and a mobile computing, we propose to overlap a wireless-mobile computing environment on an MDBS. We apply an Altruistic Locking as a transaction management and concurrency control scheme to support some problems of mobile system such as disconnection or weak connection.

표 1. 다중데이터베이스와 모바일 시스템의 비교분석
Table 1. Comparative Analysis between Multidatabase and Mobile System.

System Issues	Multidatabase	Mobile System
Heterogeneous Interoperability	Hardware and Software heterogeneity	Identical to an MDBS
Transaction Management and Concurrency Control	Correct transaction management should satisfy the ACID properties (Atomicity, Consistency, Isolation, and Durability)	Identical to an MDBS
Distribution Transparency	Distribution of the data is transparent to the user	Identical to an MDBS
System Transparency	The user should be able to access the data irrespective of the system	Identical to an MDBS
Intelligent Search and Browsing of Data	The system should provide a means for the user to efficiently search and browse the available data.	Identical to an MDBS
Location Transparency	Location of the data is transparent to the user	Identical to an MDBS
Site Autonomy	Local control over resources and data. Three different forms of autonomy include design, communication and execution.	Local control over resources and data. The degree of autonomy required depends upon the degree of mobile support offered by the system.
Location Dependence	N/A	The content of the data is physically dependent upon the location of the user and/or system.
Disconnect and Weak Connection Support	N/A	The system should provide a means to access the data while faced with a disconnection or weak connection.
Support for Resource Scarce System	N/A	The system should address the inherent limitations of various resource scarce access devices. These include processing, storage, power, and display limitations.

2. Transaction Management and Concurrency Control

Traditionally, in a distributed database system, to achieve higher performance and throughput, transactions are executed concurrently. Standard transaction scheduling schemes like two-phase locking(2PL)[11] is used for the context of concurrent execution environment in which short-lived ones are normally mixed with long-lived ones. The degree of concurrency might be hampered by selfishness associated with lock retention in 2PL. This sort of reluctance for early release of locks is essentially due to their discipline.

However, long time delay due to long-lived transactions is not good for mobile environment since transaction processing could be deferred by service handoff. Lazy release in turn could aggravate fate of misfortune for long-lived ones in that they are more vulnerable to get involved in deadlock situations. This could the other way around aggravate the fate of short-lived ones as well in a way that they suffer from starvation or livelock affected by long-lived ones. Example 1 shows this problem.

Example 1(Delay situation of short-lived transaction due to long-lived transaction): There are two types of transactions as shown in Figure 1.

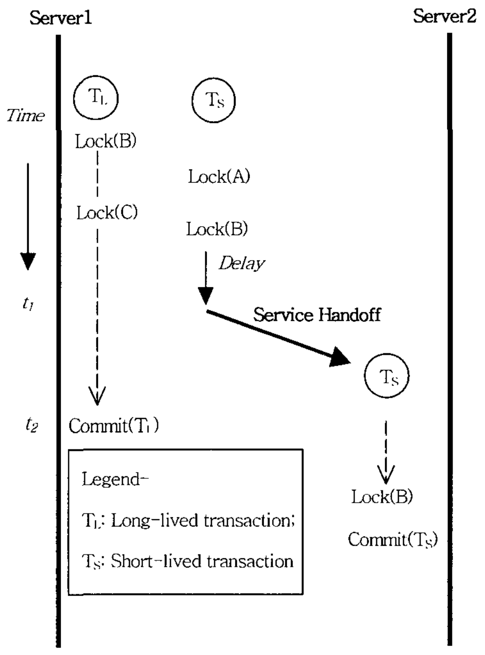


그림 1. 모바일 계산환경하에서 장기거래로 인한 대기 현상 발생 예
Fig. 1. Example of live-lock situation due to long-lived transaction in mobile computing environment.

A transaction T_L, which is long-lived, attempts to access a number of data items, including data item

B. Suppose also that a transaction T_S which is short-lived, attempts to access only two data items, say A and B. Assume that T_L has already locked B first, and T_S attempts to request lock B later on, a lock request for B from T_S shall be rejected due to lock-out of B by T_L. T_S should therefore be delayed until T_L reaches to a point where locks for all the remaining data items and then unlocks B. In other words, T_S should be delayed until T_L touched all the data it wants. T_S in this situation experiences some delay.

We assume that a service handoff occurs while T_S is delaying by T_L. The user that runs the transaction T_S, moves service area from server1 to server2. When the service handoff occurs at t_1, the context of the interactions of T_S with server1 needs to be transferred from server1 to server2 before the physical connection transfer can actually be carried out. After T_S moved to server 2, T_S can carry on its process only after T_L is committed at t_2. If execution time of T_L lasts for a long period, the delay of T_S may lead to a serious degradation of concurrency degree.

End of example 1.

Traditional concurrency control scheme like 2PL is not suitable for mobile environment since mobile could be disconnected due to a failure in communications or of a server. The context information transferring of the handoff can be a big burden for the transaction processing. Furthermore, if short transactions are out-favored, degradation of concurrency degree is inevitable. In case the degree of concurrency needs to substantially rise, for instance in mobile environment, this sort of delay effect could cause domino phenomenon. As long as long transactions and short transactions live together, we could have to live up with this kind of dilemma.

3. System Architecture and Database System for Mobile System

In general, mobile users will desire access to private and corporate database which cannot be simply geographically partitioned into locally-accessed portion. It will then be necessary to use a distributed server architecture, where the information is replicated across multiple interconnected servers but the system functions as a single logical information base. A transaction is executed at a single replica, however locks could be obtained at several sites and updates may have to be installed at several sites at the end of the transaction.

As the user moves or network load and availability changes, the server of the user may result in the virtual mobility of the server. This is accomplished by means of service handoff, which is broadly analogous to a PCS call handoff or user location update procedure[12, 13], but relatively less frequent. We have previously designed a service handoff protocol[14, 15] and described the context information which must be transferred from the old to the new server for various classes of applications, including mobile transactions.

2PL for the context of concurrent execution environment is used in example 1, and also two-phase commitprotocol is used to ensure that the transaction commits at all sites or aborts at all sites. A co-ordinator sends a prepare message to the participating replicas, upon which each replica votes whether it can commit its unit or not. If all votes are affirmative, the co-ordinator sends a commit message to the replicas, and an abort message otherwise.

Suppose that a situation of continuous mobility in which the user is running a long transaction involving the read/write operations, and the user moves between service areas while the transaction is processing. We assume that a service handoff situation in Figure 1. Whenever service handoff occurs, the user starts communicating with the local server of the new service area. The service handoff protocol maintains continuity of the physical communication link between the user and the server while the handoff occurs. However, before the physical connection transfer can actually be carried out, the context of the interactions of the user with the server needs to be transferred from the old to the new server.

Ⅲ. Proposed concurrency control scheme for mobile environment

To reduce the degree of livelock, the idea of altruism has been suggested in the literature. *Altruistic locking*[8], *AL* for short, is basically an extension to 2PL in the sense that several transactions may hold locks on an object simultaneously under certain conditions. Such conditions are signaled by an operation *donate*. Like yet another primitive *unlock*, donate is used to inform the scheduler that further access to a certain data item is no longer required by a transaction entity of that donation.

The basic philosophy behind AL is to allow long-lived transactions to release their locks early, once it has determined a set of data to which the locks protect will no longer be accessed. In this respect, effect of donate is actually to increase the degree. In order to allow more freedom, an entity of donation is let continue to acquire new locks. This implies that donate and lock operations need not be strictly two-phase.

In *AL*, the basic concept is to allow long-lived transactions to release their locks early, once it is determined that the data which locks protect will no longer be accessed.

Simulation experiments shows that *AL* outperforms the conventional *2PL* in terms of the degree of concurrency and average transaction waiting time[9]. Example 2 shows that donate operation may reduce delay time for short-lived transaction in a mobile computing environment.

Example 2(No delaying effect of short-lived transaction due to long-lived transaction): Mobile system environment and transaction scheduling is provided as same as example 1. Suppose that a transaction T_L attempts to access data item B and C and T_S attempts to access only two data items, say A and B(Figure 2).

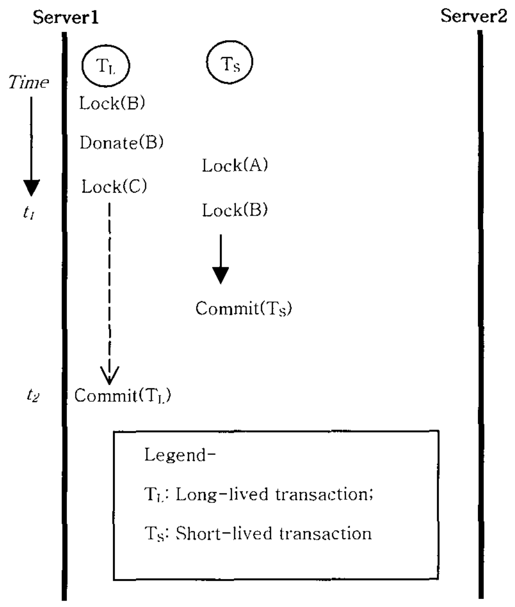


그림 2. 모바일 계산환경에서 기부연산을 이용한 동시 실행거래

Fig. 2. Concurrent transactions using donate operation in mobile computing environment.

Assume that T_L has already locked B first, and donated B later on. When T_S attempts to request lock B later on at t_1, it can be accepted successfully. Since data item B has already donated by T_L, T_S should not therefore be delayed until T_L reaches to a point where locks for all the remaining data items and then unlocks B.

T_S may have finished its process without any delay due to long transaction T_L. We can reduce the delay time due to occurring a service handoff while T_S is waiting for data item B that is already locked by T_L .

End of example 2.

The context information which need to be transferred during service handoff, thus includes the transaction id and the intentions list already done at the oldserver. This process seems to be an extra overhead of short-lived transactions. We need some different transaction concurrency control schemes except 2PL in mobile environment since the physical communication link between the user and the server could not develop successfully.

We propose *AL* as a mobile transaction concurrency control scheme since it is preferable especially for short-live transactions due to donation operation. We could reduce short transaction's delay due to a service handoff while it is waiting for some data item that was already locked by long-lived transaction.

Ⅳ. Algorithms of proposed concurrency control scheme for mobile environment

Altruism locking protocolfor mobile environment, *ALM* for short, can be pseudo-coded as follows.

1. ALM Protocol

```
Algorithm(ALM)
1. Input: LT; ST
2. /*ST:short trans;
3. LT:long trans
4. ScheduleWait():waiting action for locks being
   requested
5. ScheduleLock():locking action about requested
   data
6. ScheduleDonated():donating action about no longer
   required data*/
7. BEGIN
8. LockedObject = False
9. FOREACH LockRequest
10. FOREACH Not ServiceHandoff

11. IF(LockRequest.data = LT) THEN
12.  Call ( LockRequest_data_LT)
13. ELSE /* Is Data requested by ST? */
14. IF(LockRequest.ST.data = Lock) THEN
15. /* Locks being requested by ST already granted
to long trans */
16. Reply:=ScheduleWait(LockRequest)
17. ELSE
18. BEGIN
19. IF(LockRequest.ST.data = Donated) THEN
20.  IF (LockedObject = True) THEN
21.  Reply := ScheduleWait(LockRequest)
22.  ELSE
23.  IF ST.wake = null THEN
24.  Reply:= ScheduleDonated(LockRequest)
25.  ST.wake = ReservedLT
26. /* Acquire Lock on Requested Data */
27.  ELSE
28.  IF ST.wake = ReservedLT THEN
29. Call (Marking_set_Check)
30.  ELSE
31.  LockedObject = True
32.  Reply := ScheduleWait(LockRequest)
33.  ENDIF; /*ST.wake = ReservedLT*/
34.  ENDIF;/* ST.wake= null*/
35.  ENDIF;/* LockedObject = True*/
36. ELSE /* LockRequest.ST.data = Donated
37.  Reply := ScheduleLock(LockRequest)
38. ENDIF; /*LockRequest.ST.data =Donated*/
39. END
40. ENDIF; /*LockRequest.data = Lock*/
41. ENDIF;/*LockRequest.data = LT*/

42. IF(Reply = Abort) THEN
43. /* Lock request of ST aborted */
44. Abort Transaction(Transactionid);
45. Send(Abort);
46. Return();
47. ENDIF
48. ENDFOR/*Not ServiceHandoff*/
49. Service handoff protocol Activating
50. Context Switching between User and the
    GTM(Global Transaction Manager)
51. ENDFOR/*LockRequest*/

52. Procedure LockRequest_data_LT
53. BEGIN
54. IF(LockRequest.ST.data = Lock) THEN
55. LockedObject = True
56.  Reply:=ScheduleWait(LockRequest)
57. ELSE
58.  Reply:=Schedulelock(LockRequest)
59.  LockRequest.ST.data = Donated
60. /* Locks being requested by LT granted to long
    trans and then donated */
61. ENDIF
62. END

63. Procedure Marking_set_Check
64. BEGIN
65. IF(ST.data ∉ ReservedLT.marking-set) THEN
66. /* Data being requested by ST to be later
    accessed
67. by LT ? */
```

```
68.  Reply:=ScheduleDonated(LockRequest)
69. ELSE
70.  LockedObject = True
71.  Reply := ScheduleWait(LockRequest)
72. ENDIF
73. END

74. END
```

V. Correctness of Algorithms

We will prove that the schedule which the scheduler of *ALM* may produce is serializable in this section. To do so, we will make use of the serializability theorem[7], the definition of Crest Before[8] and a lemma used in proving the correctness of *AL*[8]. The serializability theorem states that a history H is serializable iff its serialization graph is acyclic, and the definition of Crest Before states that for two transactions, say $T_i \rightarrow_u T_j$ if T_i unlocks some data items before T_j locks some data items.

The notations used in this correctness proof are as follows. We use Tito denote the execution of either read or write operation i. $T_{ij}[x]$ denotes tothe execution of either read or write operation issued by a transaction T_i, on a data item x at server j. Locking operation for either read or write is represented by $ol_i[x]$. Unlock and donate operations are denoted by $u_i[x]$ and $d_i[x]$ respectively. A data item that is requested by transaction T_i is denoted by $o_i[x]$. H represents a history which may be produced by *ALM*. The characteristics of histories which may be produced by *ALM* are as follows.

Property 1(Two-Phase Property): If $ol_i[x]$ and $u_i[y]$ are in O(H), $ol_i[x] < u_i[y]$. This property represents two-phase rule.

Property 2(Lock Property): If oi[x] is in O(H), $ol_i[x] <$ oi[x] $< u_i[x]$. That is, a data item is locked before and unlocked after it is accessed.

Property 3(Donate Property): If $ol_i[x]$ and $d_i[x]$ is in O(H), oi[x] $< d_i[x]$. That is, T_i cannot be in its own wake.

Property 4(Unlock Property): If $d_i[x]$ and $u_i[x]$ is in O(H), $d_i[x] < u_i[x]$. That is, T_i can be unlocked after it is donated.

Property 5(Indebtedness Property): If T_j is indebted to T_i, then for every $o_j[x]$ in O(H), either $o_j[x]$ is in the wake of T_i, or there exists $u_i[y]$ in O(H) such that $u_i[y] < o_j[x]$.
If a transaction is indebted to another, it must remain completely in the other's wake until it begins to unlock objects.

Property 6(Mobility Property): If $T_{Lj}[x] < T_{Sj}[x]$ in O(H), $T_{Lk}[x] < T_{Sk}[x]$. When the service handoff occurs while T_L and T_S are working at server j on a data item x with delay situation of T_S due to T_L, T_S should be also delayed until T_L is committed at a new server k.

Lemma 1(Altruism): If $p_i[x]$ and $q_j[x]$ $(i \neq j)$ are conflicting operations in O(H) and $p_i[x] < q_j[x]$, then $u_i[x] < ql_j[x]$ or $d_i[x] < ql_j[x]$. That is, two transactions are prohibited to hold locks on the same data item unless one of them has unlocked it or donated it.
Proof: A data item must be locked before and unlocked after it is accessed by Property 2. Thus, the history, O(H), satisfies Lemma 1.
End of Lemma 1.

Lemma 2(Complexity-In-Wake): If $T_1 \rightarrow T_2$ is in serialization graph, then either $T_1 \rightarrow_u T_2$ or $T_1 \rightarrow_d T_2$.
Proof: $T_1 \rightarrow T_2$ in serialization graph means that there exist conflicting operations, say $p_1[x]$ and $q_2[x]$, in H such that $p_1[x] < q_2[x]$. There are only two cases that may occur for this by Lemma 1. One is

that there is $p_1[x] < d_1[x] < ql_2[x] < q_2[x]$ in O(H), i.e., T_2 accesses the data items donated by T_1.

A transaction T_2 has to access only wake of another transaction T_1, once T_2 makes conflict locks on the data items donated by T_1. T_2 must be completely in the wake of T_1 if T_2 has accessed any of the wake of T_1. This is ensured by the first else if condition(code line number 14) in algorithm. Even if T_2 has already accessed any data items which do not belong to the wake of T_1 , such data items would be included into the wake of T_1 as long as T_1 does not access any of such data items at all for its execution. If the data items locked by T_2 will be accessed by T_1, the access of T_2 to the data items donated by T_1 is not allowed by the second foreach condition. Thus, $T_1 \rightarrow T_2$ corresponds to $T_1 \rightarrow {}_dT_2$ in the case that $p_1[x] < d_1[x] < ql_2[x] < q_2[x]$ in H, or in the case that $p_1[x] < u_1[x] < ql_2[x] < q_2[x]$ in O(H) by Lemma 1. Thus, $T_1 \rightarrow T_2$ corresponds to $T_1 \rightarrow {}_uT_2$ in the case.

End of Lemma 2.

Lemma 3(Correctness of *AL*): Consider a path $T_1 \rightarrow \ldots T_{n-1} \rightarrow T_n$ in O(H). Either $T_1 \rightarrow {}_uT_2$, or there exists some T_i on the path such that $T_1 \rightarrow {}_uT_i$.

Proof : We will use induction on the path length n. By Lemma 2, the lemma is true for n = 2. Assume the lemma is true for paths of length n-1, and consider a path of length n. By the inductive hypothesis, there are two cases:

1) There is a T_i between T_1 and T_{n-1} such that $T_1 \rightarrow {}_uT_i$. The lemma is also true for paths of length n.
2) $T_1 \rightarrow {}_dT_{n-1}$. T_n and T_{n-1} conflicts on at least one object, x. Since T_{n-1} is completely in the wake of T_1, we must have $d_1[x] < ql_{n-1}[x]$ in O(H). By Property 2, T_n must lock x. By Property 4, T_1 must unlock x. Either $u_1[x] < ol_n[x]$ or $ol_n[x] < u_1[x]$. In the first case, we have that $T_1 \rightarrow {}_uT_n$, i.e., T_n is the T_k of the lemma. In the second case, T_n is indebted to T_1. By Property 5, T_n is completely in the wake of $T_1(T_1 \rightarrow {}_dT_n)$ or $T_1 \rightarrow {}_uT_n$.

Theorem 1(Serializability of *ALM*) : If O(H) is acyclic, O(H) is serializable.

Proof: Assume that there exists a cyclic $T_1 \rightarrow \ldots T_{n-1} \rightarrow T_n$ in serialization graph. By Lemma 2, $T_1 \rightarrow {}_dT_1$, or $T_1 \rightarrow {}_uT_i$. By Property 3, only $T_1 \rightarrow {}_uT_i$ is possible. Since T_i is prohibited to lock any more data items once T_1 unlocks any one, T_i cannot be T_1. Again, by applying Lemma 3 to the same cycle $T_i \rightarrow T_{i+1} \rightarrow \ldots \rightarrow T_i$, we get $T_i \rightarrow {}_uT_k$. For the same reason and thus we get $T_1 \rightarrow {}_uT_i \rightarrow {}_uT_k$ in all. Since the relation $\rightarrow$ u is transitive, $T_1 \rightarrow {}_uT_k$ is satisfied. Thus, T_k cannot be any of T_1 and T_i. If we are allowed to continue to apply Lemma 3 to the given cycle n-3 times more in this manner, we will get a path $T_1 \rightarrow {}_uT_i \rightarrow {}_uT_k \rightarrow {}_u \ldots \rightarrow {}_uT_m$ containing all transactions, i.e., T_1 through T_n. If we apply Lemma 3 to the given cycle starting from Tmone more time, we are enforced to get a cycle $T_1 \rightarrow {}_uT_i \rightarrow {}_uT_k \rightarrow {}_u \ldots \rightarrow {}_uT_m \rightarrow {}_uT_1$ and we get a contradiction of violating Property 1 or Lemma 3. Thus serialization graph is acyclic and by the serializability theorem O(H) is serializable.

End of Theorem 1.

Theorem 2(Mobility Satisfaction of ALM) : If H is a history with Property 6, then H satisfies mobility requirements.

Proof : By Property 5, the history of transactions is maintained even though a service handoff occurs. When the service handoff occurs while a transaction runs at its own server, the context of the interactions of transaction needs to be transferred

from old server to a new server before the physical connection transfer can actually be carried out. Let $T_{Lj}[x]$ and $T_{Sj}[x]$ be two transactions such that $T_{Lj}[x] < T_{Sj}[x]$ at server j. If $T_{Sj}[x]$ moves to a new server k while delaying due to a long transaction $T_{Lj}[x]$, $T_{S}k[x]$ should be delayed until $T_{Lj}[x]$ commits its transaction. By property 6, the history of these transactions is maintained by a global manager.
End of Theorem 2.

Ⅵ. Conclusions

We have presented a mobile computing environment for delivery of mobile transactions and identified the notion of service handoffs in this environment. We have presented an *ALM* for transaction scheduling scheme in mobile environment.

ALM is considered to be a practical solution to take in mobile environments where long-lived transactions naturally coexist with short-lived ones. Although liveness duration might not be a serious issue in the arena of standard on-line transaction processing, in which transactions are normally expected to finish shortly, it certainly matters in circumstances where a number of long-lived ones are supposed to access a substantial number of data. In traditional standard transaction scheduling schemes, such as two-phase locking, the degree of concurrency might be hampered by selfishness associated with lock retention in which service handoffs occur frequently. We in fact attemptto extend to multiple donation for short-lived transactions. It could definitely give enhance degree of freedom in accessing donated data, however, the transaction scheduler suffers from a burden in managing enlarged wakes.

References

[1] Y. Breitbart, A. Silverchatz, and G. Thompson, "An Update Mechanism for Multidatabase Systems," IEEE Data Eng. Bull., vol. 10, no. 3, pp. 12-18, Sept. 1987.

[2] Y. Breitbart, H. Garcia-Molina, and A. Silverchatz, "Overview of Multidatabase Transaction Management," Very Large Databases·J., vol. 1, no. 2, pp. 181-239, Oct. 1992.

[3] M. Bright, A. Hurson, and S. Pakzad, "A Taxonomy and Current Issues in Multidatabase Systems," Computer, vol. 25, no. 3, pp. 50-60, Mar. 1992.

[4] P. Chrysanthis, "Transaction Processing in Mobile Computing Environment," Proc. IEEE Workshop Advances in Parallel and Distributed Systems, Oct. 1993.

[5] T. Devirmis and O. Ulosoy, "Design and Evaluation of a New Transaction Execution Model for Multidatabase Systems," Information Sciences, vol. 102, pp. 203-238, 1997.

[6] P. Honeyman, L. Huston, J. Tees, and D. Bachmann, "The LITTLE WORK Project," Proc. Third IEEE Workshop Workstation Operating Systems, Apr. 1992.

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

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

[9] H. K. Rhee, "Altruism-Extensible Locking for Distributed Databases:AXL," PDPTA Proc. Parallel and Distributed Processing Techniques and Applications, Vol. 3, pp. 1457-1463, June 2000.

[10] J. B. Lim, A. R. Hurson, "Transaction Processing in Mobile, Heterogeneous Database Systems," IEEE Transactions on Knowledge and Data Engineering, Vol. 14, no. 6, Dec. 2002.

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

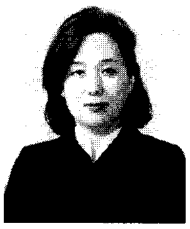
[12] "Cellular radiotelecommunications intersystem operations, Rev. B", EIA/TIA, July, 1991.

[13] M. Mouly and M. B. Pautet, "The GSM System for Mobile Communications", 49 rue Louise Bruneau, Palaiseau, France, pp.701, 1992.

[14] R. Jain, and N. Krishnakumar, "Network Support for Personal Information Services to PCS Users", IEEE Conf. Networks for Pers. Comm.(NPC), Long Branch, NJ, Mar. 1994.

[15] R. Jain, and N. Krishnakumar, "Service handoffs and virtual mobility for delivery of personal information services to mobile users", Submitted for publication, 1994.

저 자 소 개



李 惠 卿(正會員)

1979년 2월 : 숭실대학교 전자계산학과 졸업. 1985년 4월 : University of Illinois(Urbana-Champaign) 전산학과 석사. 2000년 2월 : 성균관대학교 전기전자컴퓨터공학부 박사. 1988년 3월~1989년 2월 : 국립천안공업전문대학 전자계산과 전임강사. 1992년 3월~2001년 8월 : 경인여자대학 멀티미디어정보 전산학부 조교수. 2001년 9월~현재 : 용인송담대학 컴퓨터정보과 조교수. <주관심분야 : 온라인 거래처리, 분산데이터베이스, 이동데이터베이스>