한국항행학회 논문지 제12권 제4호 2008년 8월

# 방송 환경에서 갱신위주의 이동 트랜잭션을 위한 동시성 제어 방법

# Concurrency Control for Mobile Transactions consisted mainly of Update Operations in Broadcast Environments

김치연*, 정민아**

Chi-Yeon Kim*, Min-A Jung**

## 요 약

방송은 서버와 이동 클라이언트가 무선 채널을 사용하여 효율적으로 상호작용하는 방법이며, 방송 환경은 다양한 응용들로 구체화되고 있다. 방송 환경에서 제안된 대부분의 연구들은 판독 전용 이동 트랜잭션을 주로 다루고 있으나 최근에는 이동 클라이언트에서도 갱신 트랜잭션을 수행하는 응용들이 나타나고 있다. 따라서 이 논문에서는 방송 환경에서 갱신 연산 위주로 구성된 이동 트랜잭션을 위한 동시성 제어 방법을 제안하고자 한다. 낙관적 방법으로 갱신 트랜잭션을 수행하게 되면 충돌로 인해 반복 철회되는 문제가 발생한다. 이 문제를 해결하기 위해서는 충돌 관계에 있는 트랜잭션을 분산 수행해야 하는데, 기존의 연구에서는 불필요한 철회가 발생하였다. 이에 이 논문에서는 트랜잭션을 분산 수행시키되, 초기에 발생하는 불필요한 철회를 방지할 수 있는 방법을 제안하고자 한다. 제안하는 방법은 불필요한 업링크의 사용이 없고, 이동 클라이언트의 자원을 절약할 수 있다.

## Abstract

Broadcast is a efficient interactive method between a server and mobile clients via wireless channel and broadcast environments are incarnating as various applications. Most studies have been proposed in broadcast environments deal with read-only mobile transactions, many applications are emerging recently that need to manage the update transactions at mobile clients. So we propose a concurrency control for mobile transactions consisted mainly of update operations in broadcast environments. As an optimistic approach is applied for scheduling update transactions, repetitive aborts of update transactions are occur due to conflict between transactions. To solve this problem update transactions must have been executed with distributed manner, but unnecessary aborts are occur as well because of continuous restart. Thus, in this paper we propose a method that transactions are executed distributed manner and can avoid unnecessary aborts of update transactions. Proposed method has no unnecessary uplink and can save resources of mobile client.

Key words : Update Transactions, Optimistic Concurrency Control, Restart, Conflict

* 목포해양대학교 (Mokpo National Maritime University)
** 목포대학교 (Mokpo National University)
· 제1저자 (First Author) : 김치연
· 투고일자 : 2008년 8월 7일
· 심사(수정)일자 : 2008년 8월 8일 (수정일자 : 2008년 8월 21일)
· 게재일자 : 2008년 8월 30일

## I. 서 론

서버와 클라이언트로 구성된 환경에서 데이터를 교환하는 일반적인 방법은 명시적인 요구에 의한 전달이다. 하지만 서버에 비해 비대칭적으로 많은 이동 클라이언트가 존재하는 환경에서 효율적인 데이터 전달 방법은 명시적인 요구 없이 서버가 일방적으로 데이터를 방송하는 것이다[1]. 서버는 반복적으로 데이터를 방송하며 이동 클라이언트는 방송 채널에서 필요한 데이터를 얻는다[2]. 방송을 통한 데이터 전달 방식은 경매나 주식 거래, 전자상거래 등 유무선 환경에서 다양하게 적용되고 있다.

방송 환경에서 제안된 대부분의 연구들에서는 이동 클라이언트에서 수행되는 트랜잭션을 판독 전용 트랜잭션으로 제한하고 있다[1],[3],[4]-[7]. 하지만 이동 기기의 발전과 사용자 요구의 다양화로 이동 클라이언트에서도 갱신 트랜잭션을 필요로 하는 응용들이 나타나고 있다.

방송 환경에 적합한 동시성 제어 방법은 낙관적 동시성 제어 방법이다. 이동 클라이언트의 전력의 한계는 항상 서버와 연결을 유지할 수 없으며, 2단계 잠금(2PL:2-Phase Locking)과 같은 비관적 방법을 사용하기 위해서는 잦은 메시지 교환이 발생하게 되는데, 무선 채널의 낮은 대역폭 때문에 잦은 메시지 교환은 바람직하지 않다. 판독 전용 연산으로 구성된 이동 트랜잭션의 경우에는 낙관적 방법이 비관적 방법보다 효율적이라는 것은 [8]에서 지적한 바 있다.

낙관적 방법으로 갱신 연산이 포함된 이동 트랜잭션을 수행하기 위해서는 트랜잭션 수행이 끝나면 접근한 데이터 집합을 서버에 보내 검증(Validation)을 수행한다. 이 때, 동일 방송 구간에서 충돌이 발생한 여러 트랜잭션이 있다면 정확성 기준에 의하여 하나의 트랜잭션을 제외하고는 모두 철회된다. 철회된 트랜잭션은 완료를 위해 재수행되고, 충돌이 많아질수록 검증단계에서 철회되는 트랜잭션이 많아진다.

[9]에서는 방송 환경에서 갱신 트랜잭션이 편향된 데이터 접근 패턴을 가질 때, 충돌로 인해 철회된 트랜잭션을 바로 재실행하지 않고 데이터의 경합 정도에 따라 수행을 분산시킴으로써 반복적인 재실행을 막아보고자 하였다. 하지만 이 연구에서는 철회되어 재수행되는 트랜잭션에 대하여 분산 실행을 함으로써 초기 실행에서 많은 트랜잭션이 철회되는 문제를 갖는다. 이것은 클라이언트의 자원을 낭비하는 일이다. 따라서 이 논문에서는 갱신 트랜잭션에 대하여, 접근하는 데이터의 집중 정도에 따라 처음부터 트랜잭션을 분산 수행함으로써 불필요하게 철회되지 않는 방법을 제안한다. 응답 시간의 측면에서는 [9]의 방법과 유사하나, 자원의 이용률 측면에서는 훨씬 나은 결과를 기대할 수 있다. 이 논문에서 다루는 방송 환경에 대한 시스템 모델은 그림 1과 같다.

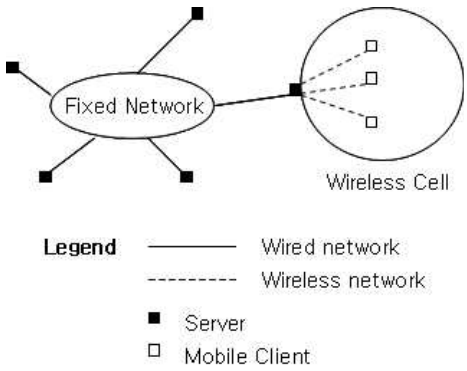



그림 1. 시스템 모델
Fig. 2. System Model

이 논문의 구성은 다음과 같다. 2장에서는 이동 환경에서 동시성 제어를 다른 기존의 연구들에 대하여 살펴보고, 3장에서는 새로운 알고리즘을 제안하고, 기존의 방법들과의 성능 비교를 다룬다. 마지막으로 4장에서는 결론을 기술한다.

## Ⅱ. 관련 연구

이 장에서는 방송 환경에서 제안된 연구들과 문제점들에 대하여 기술한다. 방송 환경에서 제안된 연구들을 분류해보면, [1]-[4],[10]에서는 서버의 방송 알고리즘을 주로 다루었고, [5],[8],[11]에서는 낙관적 동시성 제어방법에서 필요한 검증 방법과 일관성 기준을 주로 다루었다. 그리고 [6],[7],[9]에서는 동시성 제어 알고리즘을 주로 다루었다.

[6]에서는 push-based 방송과 pull-based 방송을 복합적으로 사용하는 환경에서 이동 트랜잭션의 처리

방법을 제안하였다. 갱신 트랜잭션은 서버에서만 수행하도록 제한하였으며, 낙관적 방법과 비관적 방법을 혼합한 동시성 제어 방법을 제안하였다. 이 연구는 이동 클라이언트에서 갱신을 처리할 수 없을뿐더러, 충돌이 발생되어 비관적 방법으로 트랜잭션을 처리하게 되면, 이동 클라이언트에서 많은 메시지 전송을 필요로 한다는 문제점이 있다.

[9]에서는 방송 환경에서 이동 클라이언트가 갱신 트랜잭션을 처리할 수 있을 때, 갱신 데이터가 특정 데이터에 집중되는 경우의 낙관적 동시성 제어 방법에 대하여 기술하였다. 이 연구의 문제점을 기술하기 위해 순수한 낙관적 방법으로 편향된 갱신 트랜잭션을 수행하는 경우와, [9]에서 제안된 방법의 수행 예를 기술하면 그림 2와 같다.

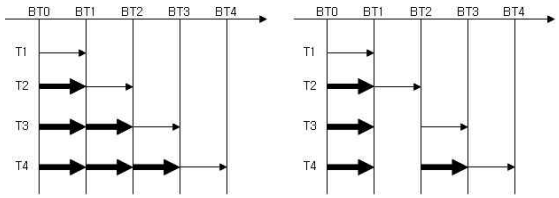



그림 2. 낙관적 방법의 트랜잭션 수행 예
Fig. 2. Example of Transaction Execution with Optimistic Approach

그림 2는 네 개의 트랜잭션 T1 ~ T4가 방송 구간 BT0 ~ BT4까지 수행되는 형태를 간단하게 보여주는데, 네 개의 트랜잭션의 모두 공통된(집중된) 데이터를 갱신하는 경우의 수행 예이다. 네 개의 트랜잭션은 같은 데이터를 갱신하기 때문에 하나의 방송 구간에서는 하나의 트랜잭션만 완료할 수 있다. 왼쪽 그림은 전형적인 낙관적 방법으로 수행한 예인데 모든 트랜잭션이 완료하기까지 여섯 번의 반복적인 철회가 발생한다. 오른쪽 그림은, [9]에서 제안한 방법으로 철회된 트랜잭션에 대하여 트랜잭션을 분산 수행함으로써 트랜잭션의 반복적인 재실행은 줄었으나, 초기에 모든 트랜잭션이 실행을 동시에 시작함으로써 하나의 트랜잭션을 제외한 나머지 트랜잭션은 일단 철회된 후 재실행된다. 모든 트랜잭션이 완료되기까지 네 번의 철회가 발생한다. 트랜잭션의 수행 초기에 발생하는 철회는 불필요한 철회라고 할 수 있다.

제안하는 방법에서 트랜잭션이 수행되는 과정을 유사한 그림으로 표현하면 그림 3과 같다. [9]의 방법과 비교해보면, 트랜잭션은 접근하는 데이터의 집중 정도에 따라 처음부터 분산 실행된다. 충돌 관계에 있는 트랜잭션을 처음부터 동시에 실행하지 않고, 데이터의 집중 정도를 이용하여 분산 실행함으로써 불필요하게 철회되는 트랜잭션 수를 줄일 수 있다.

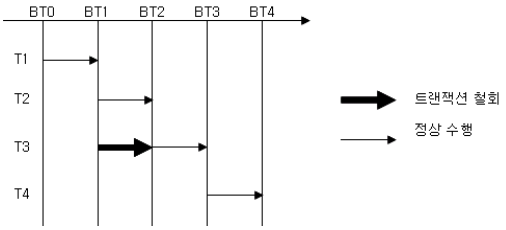



그림 3. 제안하는 방법의 트랜잭션 수행 예
Fig. 3. A Example of Transaction Execution with Proposed Approach

제안하는 알고리즘을 기술하기에 앞서 제안하는 방법에서 사용하고 있는 가정은 다음과 같다.

- 이동 클라이언트는 캐쉬를 사용한다.
- 서버에서는 주기적인 방송을 통해 갱신된 데이터를 이동 클라이언트에게 알려준다.
- 이동 트랜잭션은 하나의 방송 주기 안에서 수행을 마친다.
- 철회 후 재실행되는 트랜잭션은 접근 불변성에 의해 철회되기 전 트랜잭션과 접근하는 데이터 항목은 동일하다.
- 이동 트랜잭션은 시작하기 전 미리 접근할 데이터를 파악할 수 있다.

## Ⅲ. 제안하는 알고리즘

이 장에서는 제안하는 이동 클라이언트 알고리즘과 서버 알고리즘, 그리고 제안하는 방법의 정확성 증명 순으로 기술한다. 편의상 제안하는 방법을 CCB-UA(a Concurrency Control for Broadcast environments without Unnecessary Aborts) 방법이라 명명한다.

이 논문에서 이동 트랜잭션이라 함은 이동 클라이언트에서 수행되는 트랜잭션을 의미하며, 갱신 트랜

잭션은 갱신 연산이 포함된 트랜잭션을 의미한다. 이 논문에서 갱신 트랜잭션은 서버와 이동 클라이언트에서 수행가능하다. 판독 전용 트랜잭션은 판독 연산만으로 구성된 트랜잭션을 의미한다.

### 3-1 이동 클라이언트 알고리즘

이동 클라이언트의 첫 번째 기능은 서버와의 잦은 연결 없이 데이터를 접근할 수 있는 캐쉬의 일관성을 유지하는 것이다. 이동 트랜잭션이 접근할 데이터가 캐쉬에 없다면 클라이언트는 서버에 요청한다. 캐쉬에 있는 데이터는 서버의 주기적인 방송을 통하여 갱신되고, 각 데이터 항목에 대하여 집중값(CV : Contention Value)의 사본을 유지한다. 집중값은 방송 메시지에 포함된다. 이동 클라이언트가 단절 후 재연결되는 경우는 다음 방송을 기다려 방송 메시지에 따라 캐쉬를 갱신한 후 수행을 시작한다.

두 번째 기능은 이동 트랜잭션 $MT_{ij}$를 시작하기 전 갱신할 데이터의 집합인 $UpdateSet_{ij}\{\ \}$을 구하고, 데이터의 집중값에 따라 대기 사이클 WCycle을 결정하는 것이다. 여기서 i는 이동 클라이언트의 식별자이고, j는 트랜잭션 식별자이다. 식별자 부여 방법에 대해서는 별도로 기술하지 않는다. WCycle은 $UpdateSet_{ij}\{\ \}$의 각 집중값 중 가장 큰 값 CV에 대하여 0 ~ (CV-1) 사이의 임의의 값으로 결정되며 현 방송 구간에서 WCycle만큼 대기 후에 트랜잭션을 시작한다. 대기 사이클은 트랜잭션 수행을 분산시켜 불필요한 철회를 막기 위해 필요하다.

세 번째 기능은 이동 트랜잭션을 수행하는 기능이다. 트랜잭션의 각 연산은 데이터가 캐쉬에 있는 한 캐쉬된 데이터를 이용하여 수행되며, 갱신 트랜잭션의 경우 마지막 연산 수행 후 검증을 위해 서버에 $UpdateSet_{ij}\{\ \}$과 $AbortCounter_{ij}$를 보낸다. $AbortCounter_{ij}$는 트랜잭션 $MT_{ij}$가 철회된 횟수로 CV 값을 조정하기 위해 필요하다.

이동 클라이언트 알고리즘에서 서버와의 메시지 교환이 발생하는 경우는 캐쉬에 없는 데이터를 요청하는 경우와 갱신 트랜잭션의 검증 정보를 서버에 보내는 경우이다. 이 두 가지 메시지는 동시성 제어를 위해 반드시 필요한 정보이다. 심지어 이동 클라이언트가 단절 후 재연결되는 경우, 이동 클라이언트에서 메시지를 보내 서버의 정보를 요구할 수도 있으나, 이 알고리즘에서는 다음 방송을 기다림으로써 부가적인 메시지 교환을 피하고 있다. 따라서 제안하는 알고리즘에서 동시성 제어를 위해 서버와 클라이언트 사이에서 부가적으로 필요한 업링크는 없다고 할 수 있다.

방송 사이클 BCk동안 이동 클라이언트 $MC_i$에서 제출된 이동 트랜잭션 $MT_{ij}$를 위한 알고리즘은 그림 4와 같다.

```
// Maintain Cache Consistency
On receiving BM  // Broadcast Mesg
  update its cache
// Evaluate CV
UpdateSet_ij{ } = all data item x of w_ij(x)
WCycle = Random(Max(CV(UpdateSet_ij{ }) - 1
wait for BC(k + WCycle)th cycle
// Execute op(x)
for each operation op_ij(x) of MT_ij
  if there is no x in its cache then
    send a data request server
for each operation r_ij(x) of MT_ij
  return x
for each operation w_ij(x) of MT_ij
  update x
// Commit and Validation
if op_ij(x) is a last operation of MT_ij then
  if MT_ij is a ROT then // Read-Only Tr.
    commit MT_ij
  if MT_ij is a UT then  // Update Tr.
    send UpdateSet_ij{ } and AbortCounter_ij a
        Server
    wait for a Decision
//Maintain a AbortCounter
if Decision = Abort then
  AbortCounter_ij ++
// Reconnection mode
if (on reconnection mode) then
  wait for the next BM
```

그림 4. 이동 클라이언트 알고리즘
Fig. 4. A Algorithm for Mobile Clients

### 3-2 서버 알고리즘

서버의 첫 번째 기능은 방송 메시지의 구성과 방송이다. 서버는 하나의 방송 구간동안 완료된 트랜잭션이 갱신한 값과 데이터의 집중값으로 방송 메시지를 구성하여 이동 클라이언트들에게 방송한다. 데이터의 집중값 CV는 충돌로 인해 트랜잭션이 철회될 때마다 증가되고, 재수행 트랜잭션이 완료될 때 트랜잭션의 AbortCounter만큼 감소된다. 방송 디스크에 포함되는 데이터는 플랫한 형태가 아닌 접근 빈도에 따라 멀티디스크의 형태로 제공된다[2]. 즉, 하나의 방송 구간동안 접근 빈도가 높은 데이터는 그렇지 않은 데이터에 비해 자주 방송되며, 인터리빙된 형태로 제공된다. 또한 방송의 초기에 각 데이터 항목을 위한 인덱스를 제공하여 이동 클라이언트가 계속적으로 방송 메시지를 청취하지 않아도 데이터 항목을 수신할 수 있도록 한다. 데이터 A의 접근 확률이 B, C의 두 배일 때 방송 메시지의 예는 그림 5와 같다.

두 번째 기능은 갱신 트랜잭션에 대한 검증기능이다. 검증은 현재 수행중인 트랜잭션의 연산들과 충돌을 검사하는 정방향 검증(Forward Validation)과, 방송 구간 BT에서 검증 요청 전까지 완료된 트랜잭션의 갱신 집합 $CommitSet_{BT}\{\ \}$와 교집합을 검사하는 역방향 검증(Backward Validation)을 모두 수행한다. 정방향 검증에서 충돌이 탐지되면 수행 중인 트랜잭션이 철회되며, 역방향 검증에서 충돌이 탐지되면 검증 단계의 트랜잭션이 철회된다. 두 검증에 모두 성공하면 트랜잭션은 완료되며, 다음 방송 메시지에 갱신 값이 포함된다. 서버를 위한 알고리즘은 그림 6과 같다.

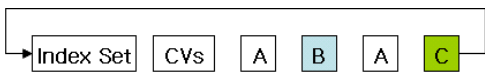



그림 5. 방송 메시지 구조
Fig. 5. Structure of Broadcast Message

```
// Broadcasting
On Broadcasting Time BT, construct a BM
    and broadcast to all Mobile Clients
Clear CommitSet_BT{ }
// Validation Processing
if a Update Transaction T_ij requests a validation
  result = Validation(T_ij)
if result = Abort then
  for each data x of w_ij(x)
    contention(x) += 1
if result = Commit then
  for each data x of w_ij(x)
    contention(x) -= AbortCounter_ij
    CommitSet_BT{ } = CommitSet_BT{ } ∪{x}
Validation(T_ij)
for each operation w_ij(x) of T_ij
  if there is conflict with current active op_k(x) of
     Server Transaction T_k then
       Abort T_k
  else if (CommitSet_BT{ } ∩
       UpdateSetSet_ij{ } ≠ ∅)
    then return Abort
    else return Commit
```

그림 6. 서버 알고리즘
Fig. 6. Algorithm for a Server

다음은 제안하는 CMB-UA 방법을 적용한 트랜잭션의 수행 예를 기술한다. 갱신 위주의 이동 트랜잭션이므로 모든 이동 트랜잭션의 갱신 연산 비율은 50% 이상으로 간주한다. 이동 트랜잭션은 제출되면 곧바로 수행되며, 한 방송 주기 안에 수행을 마친다. 예에서 MT는 이동 트랜잭션을 의미하며 ST는 서버 트랜잭션을 의미한다. 데이터 항목 a, b, c, d, e의 초기 집중값은 각각 1, 1, 2, 2, 0이고, 네 개의 트랜잭션이 모두 BT0에서 제출되었다고 가정한다.

초기 집중값 CVs a:1 b:1 c:2 d:2 e:0

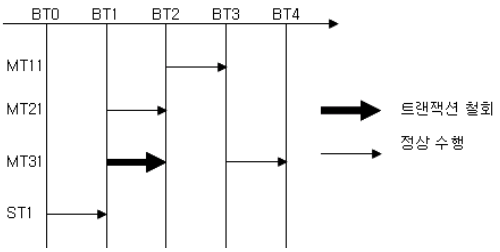
$MT_{11}$ : $r_{11}(b)$ $w_{11}(c)$ $w_{11}(e)$

$MT_{21}$ : $w_{21}(a)$ $r_{21}(d)$ $w_{21}(b)$

$MT_{31}$ : $w_{31}(a)$ $w_{31}(b)$ $r_{31}(c)$ $w_{31}(d)$

$ST_1$ : $r_1(e)$ $w_1(c)$ $r_1(d)$ $w_1(a)$

| 트랜잭션 | UpdateSet{} | CVs | WCycle |
|---|---|---|---|
| $MT_{11}$ | {c,e} | {2,0} | 2 |
| $MT_{21}$ | {a,b} | {1,1} | 1 |
| $MT_{31}$ | {a,b,d} | {0,1,2} | 1 |
| $ST_1$ | {c,a} | | 0 |


방송 시점 BT0 이후에 서버 트랜잭션 $ST_1$이 제출되어 성공적으로 완료한다. 방송 시점 BT1의 방송 메시지에는 $ST_1$이 갱신한 데이터 c, a에 대한 갱신 값이 포함된다. BT1 이후에는 WCycle 값만큼 대기한 $MT_{21}$과 $MT_{31}$이 함께 제출되어 수행되는데 이는 각각 다른 이동 클라이언트에서 수행되므로 서버에 검증 단계를 거치기 전까진 충돌 관계를 알 수 없다. $MT_{21}$이 먼저 검증을 시작하고 성공적으로 완료하면, 두 트랜잭션의 충돌 관계로 인하여 $MT_{31}$은 철회되고, $MT_{31}$은 다시 대기 사이클만큼(예에서는 1) 대기에 들어간다. $MT_{21}$이 완료되면 $MT_{21}$이 갱신한 데이터를 반영하는데, $MT_{21}$은 재수행없이 수행된 트랜잭션이므로 $AbortCounter_{21}$은 0을 갖고 따라서 CV 값들은 변경되지 않는다. $MT_{31}$이 철회되면 a, b, d의 CV 값은 하나 증가한다. 다음 방송 구간에서는 2개의 사이클을 기다린 $MT_{11}$이 제출되어 완료한다. $MT_{11}$도 재수행 트랜잭션이 아니므로 CV 값에는 영향을 미치지 않는다. 마지막 방송 구간에서 $MT_{31}$이 제출되어 수행되므로 성공적으로 완료되면 트랜잭션이 접근한 a, b, d의 CV 값을 $AbortCounter_{31}$ 값인 1만큼 감소시킨다.

### 3-3 정확성 증명 및 성능 비교

이 절에서는 제안하는 방법의 정확성을 증명하기 위하여 제안하는 방법에 의해 생산되는 히스토리가 직렬가능한 히스토리[12]인지를 증명한다.

1) 판독 전용 트랜잭션의 직렬성 유지

이 논문에서는 주로 갱신 트랜잭션을 다루고 있으나 이동 클라이언트의 특성상 판독 전용 트랜잭션의 수행 가능성도 배제할 수 없다. 판독 전용 트랜잭션은 모든 연산이 판독 연산만으로 구성되는데, 제안하는 알고리즘에 의하면 판독 연산은 이동 클라이언트의 캐쉬를 접근하여 수행되며, 서버 검증을 거치지 않고 자체적으로 완료할 수 있다. 이 경우의 직렬가능성에 대하여 증명한다.

제안하는 CCM-UA 알고리즘이 직렬성을 유지하지 못한다면 직렬화 그래프 SG에 사이클이 발생된다. 즉, $T_i \rightarrow T_j \rightarrow \cdots \rightarrow T_i$와 같은 사이클이 생기게 된다(여기서 T는 이동 트랜잭션이거나 서버 트랜잭션이다). 직렬화 그래프에 간선이 추가되는 경우는 동일 데이터를 접근하는 두 연산 중 하나가 기록(write)연산인 경우이다. 즉, r-w, w-w, w-r중 한 가지 경우이다. 그림 7에서 $T_1$, $T_2$는 갱신 트랜잭션이고 $T_3$은 판독전용 트랜잭션이다. $T_1$은 이전 방송 구간에서 수행을 완료하였고, $T_2$와 $T_3$은 동일 방송 구간에서 수행되고 있다. 문제가 발생할 수 있는 히스토리 H3과 H4에 대하여 직렬화 순서를 살펴보면, H3의 경우, $R_3$(x)에서 접근하는 x의 값은 read-from 관계에 의해 $T_2$가 갱신한 값이 아니라 $T_1$이 기록한 값이다. $T_2$가 기록한 값을 $T_3$이 읽기 위해서는 $T_2$는 $T_3$보다 선행하는 방송 구간에서 수행되어 완료되어야 하는데, 이것은 모순이다. 마찬가지 원리로, H4에서 $R_3(y)$가 접근하는 y값은 $T_2$가 아닌 $T_1$이 갱신한 값이다. 따라서 가능한 히스토리 어떤 경우에서도 사이클은 찾을 수 없기 때문에 제안하는 방법에 의해 판독 전용 이동 트랜잭션이 만들어내는 히스토리는 직렬가능하다고 할 수 있다.

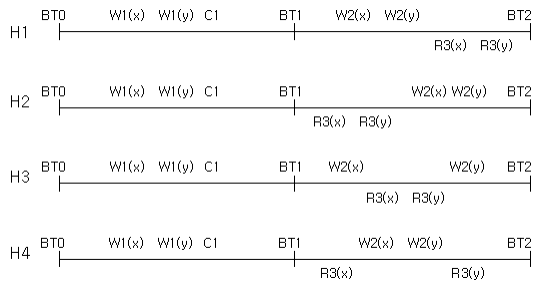



그림 7. 판독전용 트랜잭션의 직렬성
Fig. 7. Serializability of Read-only Transactions

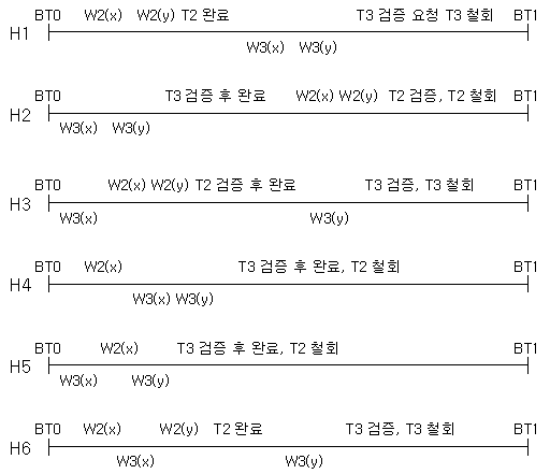



그림 8. 갱신 트랜잭션의 직렬성
Fig. 8. Serializability of Update Transactions

2) 갱신 트랜잭션의 직렬성 유지

제안하는 CCM-UA 방법에서는 모든 트랜잭션이 하나의 방송 주기 안에서 수행을 마친다고 가정하였다. 따라서 임의의 트랜잭션이 제출되기 이전 방송 구간에서 완료된 트랜잭션과의 수행 순서에는 사이클이 발생하지 않는다는 걸 알 수 있다. 고려해야 할 관계는 하나의 방송 주기 동안 수행되는 충돌 트랜잭션들 사이의 순서이다. 그림 8에서 $T_2$는 서버 트랜잭션, $T_3$은 이동 트랜잭션이라 가정하자. 여섯 개의 가능한 히스토리에서 하나의 트랜잭션은 알고리즘에 의해 반드시 철회되는데, 이로써 직렬화 그래프에는 사이클이 발생하지 않고, 직렬화 순서는 유지된다.

다음으로는 제안하는 방법을 유사한 연구인 [9]에서 제안한 AOCCRB 방법과 트랜잭션의 철회율의 측면에서 비교하고자 한다. [9]에서는 트랜잭션을 분산 수행하지 않은 기존의 낙관적 방법과 분산 수행한 경우의 응답 시간, 하향 대역폭, 상향 대역폭의 기준으로 분산 수행한 경우가 더 나음을 증명하였으므로 이 논문에서는 생략한다.

AOCCRB 방법에서는 트랜잭션이 제출되면 일단 수행된다. 동시에 제출된 트랜잭션 수를 n이라 가정하고, 모두 충돌이 있다고 가정한다면, 첫 phase에서 1-(1/n)만큼의 비율로 항상 철회가 발생한다. 즉, 네 개의 트랜잭션이 제출되었다면 하나를 제외를 제외한 세 개의 트랜잭션은 항상 철회된다.

하지만 제안하는 방법에서는 첫 번째 phase부터 트랜잭션을 분산 수행하므로 트랜잭션이 철회될 확률은 0 ~ (1-(1/n))이다. 즉, 최악의 경우에 AOCCRB와 동일하다. 트랜잭션의 철회율이 줄어들수록 트랜잭션의 응답 시간은 빨라지고, 재수행으로 인한 자원의 낭비를 막을 수 있다. AOCCRB 방법보다 제안한 CCM-UA 방법이 명백히 철회율이 낮으므로 그 차이만큼 반복 수행으로 인한 이동 클라이언트의 자원이 낭비되는 것도 줄일 수 있다.

## Ⅳ. 결 론

방송 환경에서 동시성 제어를 다룬 대부분의 연구들에서는 판독 전용 이동 트랜잭션을 주로 다루었다. 하지만 전자상거래와 같은 응용이 이동 클라이언트에서 수행가능하게 됨으로써 이동 클라이언트에서 실행되는 갱신 트랜잭션의 관리 방법이 필요하게 되었다. 갱신 트랜잭션의 수행을 위해 방송 환경에서 주로 사용되는 낙관적 방법을 갱신 트랜잭션의 동시성 제어에 적용하면 잦은 충돌과 연속적인 재수행으로 인해 트랜잭션의 철회가 많이 발생한다. 따라서 이 논문에서는 이동 클라이언트가 갱신 트랜잭션을 주로 수행하는 환경에서, 충돌로 인해 발생하는 잦은 철회와 재수행의 문제를 해결하기 위한 알고리즘을 제안하였다.

갱신 트랜잭션의 동시성 제어를 다룬 유사한 연구에서는 트랜잭션을 일단 수행한 후, 재수행 단계에서 데이터 집중도에 따라 트랜잭션을 분산 수행하였다. 하지만 초기에 트랜잭션들이 동시에 접근하는 데이터가 있는 경우, 하나의 트랜잭션을 제외한 나머지 모든 트랜잭션들은 철회된다. 이것은 불필요한 철회이다. 따라서 제안하는 알고리즘에서는 갱신 트랜잭

션이 접근하는 데이터의 집중도에 따라 처음부터 트랜잭션을 분산 수행하였다. 이렇게 함으로써 기존의 연구에서 발생하는 초기의 불필요한 철회를 방지할 수 있었고, 데이터 집중도 범위 내에서 트랜잭션들을 분산 수행하므로 연속적인 철회도 방지할 수 있었다. 또한 검증 정보를 보내거나 캐쉬에 없는 데이터를 요청하는 경우를 제외하고는 동시성 제어를 위해 서버와 불필요하게 메시지를 주고받는 일이 없어, 부가적인 업링크의 사용이 없고, 트랜잭션 철회를 줄임으로써 이동 클라이언트의 자원을 낭비하는 일도 줄일 수 있었다.

제안하는 방법의 성능을 좌우하는 중요한 요소는 충돌 관계의 트랜잭션을 겹치지 않게 방송 구간에 분산시키는 방법이다. 갱신 연산이 많다고 해도, 충돌 관계의 트랜잭션이 동시에 수행되지 않는다면 성공적으로 완료될 수 있기 때문이다. 이 논문에서는 이동 클라이언트에서 접근하는 데이터의 집중 값에 따라 임의적으로 대기 사이클을 결정했지만, 서버에서 트랜잭션이 겹치지 않게 대기 사이클을 설정해 이동 클라이언트들에게 전달한다면 좀 더 나은 성능을 기대할 수도 있을 걸로 예상된다. 따라서 보다 효율적으로 대기 사이클을 결정하는 방법과 여러 방송 구간 동안 수행되는 트랜잭션의 제어 방법은 좀 더 연구해야 할 것으로 생각된다.

## 감사의 글

이 논문은 목포해양대학교 2007년도 기성회 예산의 지원에 의함.

## 참 고 문 헌

[1] S. Acharya, M. Franklin, S. Zdonik, "Disseminating Updates on Broadcast Disks," *Proc. of 22nd VLDB Conference*, pp. 354-365, India, 1996.

[2] Pictoura, E. and Chrysanthis, P. K., "Scalable Processing of Read-only Transactions in Broadcast Push," *Proc. of the 19th IEEE International Conference on Distributed Computing System,* pp. 432-439, 1999.

[3] Nam, SungHun ; Chung, IlYoung ; Hwang, Chong-Sun, "An Efficient Cache Invalidation Scheme for Mobile Wireless Environments," *Proc., Eighth International Conference on Parallel and Distributed Systems*. pp. 26-29 June 2001.

[4] S. Acharya, M. Franklin, S. Zdonik, and R. Alonso, "Broadcast Disks : Data Management for Asymmetric Communication Environments," *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 199-210, 1995.

[5] Srinivasa, R., and Son, S, H., and Hansson, J., "Quasi-consistency and Caching with Broadcast Disks," *Proc. of Second International Conference on Mobile Data management*, pp. 133-144, 2001.

[6] 김성석, 양순옥, "복합 브로드캐스팅 환경에서 이동 트랜잭션 처리," *정보과학회 논문지 :* 데이터베이스 제31권 제4호, pp. 422-431, 2004.

[7] 임종원, 황병연, "이동 컴퓨팅 환경에서 데이터의 주기성을 고려한 캐쉬 일관성 기법," *멀티미디어학회논문지,* Vol. 10, No. 4, pp. 421-431, 2007.

[8] Lee, V.C.S, Kwok-Wa Lam, Son, S.H., "On Transaction Processing with Partial Validation and Timestamp Ordering in Mobile Wireless Environments," *IEEE Transactions on Computers,* Vol. 51, No. 10, pp. 1196-1211, 2002.

[9] 정성원, 박성근, 최근하, "무선 브로드캐스트 환경에서 편향된 엑세스 패턴을 가진 모바일 트랜잭션을 위한 효과적인 동시성 제어 기법," *정보과학회논문지* : 데이터베이스 제33권 제1호, pp. 69-85, 2005.

[10] Wu, Simon, Lee, V.C.S. and Lam, Kwok-wa "Broadcast Transaction Scheduling in Mobile Computing Environments," *Proc. of the 3rd International Conference on Mobile Data Management*, pp. 161-162, Singapore, Jan.,

2002.
[11] Das, A. and Kai, K. Y., "Tradeoff between Client and Server Transaction Validation in Mobile Environments," *International Symposium on Database Engineering & Application,* pp. 265-272, 2001.
[12] P. A. Bernstein, V. Hazilacos, N. Goodman, "Concurrency Control and Recovery in Database Systems," *Addison Wesley,* 1987.

## 김 치 연 (金致連)

1992년 : 전남대학교 전산통계학과(이학사)
1994년 : 전남대학교 전산통계학과 (이학석사)
1999년 : 전남대학교 전산통계학과 (이학박사)
2002년 3월~현재 : 목포해양대학교 해양전자통신공학부 교수
관심분야 : 이동 컴퓨팅, 전자상거래, 트랜잭션 관리, 유비쿼터스 컴퓨팅

## 정 민 아 (鄭玟婀)

1992년 : 전남대학교 전산통계학과 (이학사)
1994년 : 전남대학교 전산통계학과 (이학석사)
2002년 : 전남대학교 전산통계학과 (이학박사)
2002년 ~ 2003년 광주과학기술원 정보통신학과 박사후과정
2005년 ~ 현재 : 목포대학교 컴퓨터교육과 교수
관심분야 : 데이터베이스, 데이터마이닝, 생물정보학, 정보보호