이동 컴퓨팅 환경에서 데이타 방송을 위한 동시성 제어 기법

(A Concurrency Control Method for Data Broadcasting in Mobile Computing Environment)

윤 혜 숙 † 김 영 국 ††

(Hyesook Yoon) (Young-Kuk Kim)

요 약 수많은 이동 클라이언트가 있는 이동 사용자 환경에서 데이타 방송 기법은 매우 효과적인 데이타 전달 방식으로 주목을 받고 있다. 이 방식에서 데이타베이스 서버는 데이타를 무선채널을 통해 주기적으로 배포하며 클라이언트는 필요한 데이타를 선택적으로 액세스하는 읽기 전용 트랜잭션을 수행한다. 한편, 서버에서는 데이타 방송과 병행해서 데이타베이스 갱신도 수행하므로 클라이언트가 일관성 있는 데이타를 액세스할 수 있으려면 동시성제어 문제가 해결되어야 한다. 본 연구에서는 이러한 동시성제어 문제를 효율적으로 해결하는 알고리즘인 SCDSC(Serialization Checking with DirtySet on Commit) 기법을 제안한다. SCDSC는 이동 클라이언트에서 다중 데이타를 요구하는 읽기 트랜잭션을 커미트할 때 일반 데이타와 함께 방송된 DirtySet을 점검하여 일관성을 유지하는 일종의 낙관적 동시성 제어기법이다. DirtySet은 일정 방송주기 동안 변경된 데이타 집합으로 방송주기가 바뀔 때마다 슬라이딩 윈도우 방식으로 서버에서 갱신되어 배포된다. 또한, 제안하는 알고리즘의 성능을 데이타 일관성(data consistency) 및 현재성(data currency) 관점에서 분석하고 시뮬레이션을 통해 알아본다.

키워드 : 이동 데이타베이스, 데이타 일관성, 데이타 현재성, 데이타 방송, 동시성 제어

Abstract Data broadcast has received much attention as a very efficient method for disseminating data items in mobile environment with large number of mobile clients. In this approach, a database server periodically and continuously broadcasts data items through wireless channels and clients perform read-only transactions by accessing necessary data items from the air. While broadcasting, the server must also process update transactions on the database, which raises an obstacle for client's accessing consistent data. In this research, we propose a new algorithm SCDSC(Serialization Checking with DirtySet on Commit) which is an alternative for solving the concurrency control problem efficiently. The SCDSC is a kind of optimistic concurrency control in that a client checks the consistency of data using a DirtySet as a part of data broadcast when it commits its transaction. In each broadcast cycle, the server updates and disseminates the DirtySet with newly changed data items for last few cycles in the sliding window approach. We perform an analysis and a simulation study to evaluate the performance of our SCDSC algorithm in terms of data consistency and data currency.

Key words : mobile database, data consistency, data currency, data broadcast, concurrency control

1. 서 론

무선 기술은 빠르게 대중화되고 있다. 수백만의 이동 사용자가 작은 PCS 폰이나 팜탑을 가지고 다니며 끊임없이 증권 정보, 교통 정보, 지역 정보, 기상 정보 등을 이용한다. 무선 매체는 전 지역에 걸쳐 상당한 양의 정보를 액세스하고 조직하는 정보 고속도로의 첨병으로 무선 통신 채널을 통해 데이타를 얻는다.

서버에서 클라이언트로 데이타를 제공하는 방식에는 요청에 의한(on-demand) 기법과 데이타 방송(data broadcast)기법[1,2]의 두 가지 기법이 있다. 클라이언트가 요구할 경우 데이타가 서버로 전달되는 기존의 on-demand 기법은 수많은 이동사용자 환경에 더 이상

· 본 연구는 ITRC 및 충남대학교 학술진흥장학재단의 지원을 받았음

† 비 회 원 : 충남대학교 전기정보통신공학부 교수
hsyoon@cs.cnu.ac.kr
†† 정 회 원 : 충남대학교 전기정보통신공학부 교수
ykim@cs.cnu.ac.kr

논문접수 : 2002년 9월 9일
심사완료 : 2004년 2월 20일

적합하지 않다. 무선 채널은 대역폭이 제한되어 있을 뿐 아니라 서버에서 클라이언트로의 하향 링크(down-link)보다 클라이언트에서 서버로의 상향 링크(up-link) 대역폭이 훨씬 작은 비대칭 구조이기 때문이다. 따라서 여러 클라이언트에서 데이타를 요구하는 트랜잭션들이 발생한다면 상향 링크의 혼잡과 동일한 데이타의 중복 전송으로 데이타 대기 시간은 매우 길어질 것이다. 또 다른 방법인 데이타 방송 기법은 특별히 요청하지 않아도 서버가 주기적으로 데이타를 클라이언트로 배포하며, 이동 클라이언트에서 발생한 트랜잭션은 필요한 데이타를 무선 채널 상에 방송 중인 데이타 중에서 선택적으로 액세스한다[1,3-5]. 일반적으로 같은 데이타 아이템에 대한 많은 클라이언트가 존재할 것이기 때문에 데이타 배포 비용은 클라이언트 수와 무관하며 제한된 대역폭을 훨씬 더 유용하게 사용할 수 있다. 최근의 응용은 대규모 클라이언트 집단에 정보를 배포해야 하는 경우가 대다수이므로 이러한 방송 모드가 적합하다.

그러나 클라이언트가 서버에게 직접 요청하여 얻는 데이타에 비해 방송을 통해 데이타를 얻을 때는 동시성 제어 문제가 있을 수 있다. 즉, 방송 서버가 특정 데이타를 방송하고 있으면서 동시에 데이타베이스 서버에서 그 데이타에 대해 갱신 트랜잭션을 수행할 수 있다는 것이다. 어떤 클라이언트가 이 데이타를 수신하면 데이타의 일관성(data consistency)이 깨어질 수 있으며 데이타 현재성(data currency)도 보장할 수 없다. 데이타 일관성은 트랜잭션에서 요구하는 데이타가 무결성 제한조건이 침해되지 않는 데이타베이스 상태에 있을 때 유지된다. 이것은 클라이언트가 특정 트랜잭션을 수행할 때 그 트랜잭션에 소요되는 데이타 아이템들이 같은 갱신 주기 내에 있을 때 보장할 수 있다. 또 데이타 현재성은 클라이언트에서 발생한 트랜잭션에 의해 커미트된 값이 서버에서의 값과 일치해야 함을 의미한다. 이동 컴퓨팅에서 많은 데이타 아이템은 특정 도로의 현재 교통조건, 특정 도시의 현재 기상 상태 및 특정 주식의 최신 시세와 같은 실시간 정보를 반영해야 하기 때문에, 그 값은 상당히 동적이고 민감하다. 이미 옛 버전이 된 데이타 아이템을 액세스하는 것은 이동 클라이언트에 있는 정보의 유용성에 심각한 영향을 줄 수 있어 바람직하지 않다[6]. 서버는 계속해서 가장 최신 정보로 갱신하기 위해 데이타베이스에 있는 데이타 아이템 값을 교체하고 이를 방송할 필요가 있다[7]. 결과적으로 서버에서 갱신 트랜잭션과 데이타 방송을 병렬로 실행할 수 있으려면, 동시성제어 문제가 해결되어야 한다. 2단계 잠금(2PL)이나 낙관적 기법(OCC)과 같은 기존의 동시성제어 프로토콜은 클라이언트의 수가 많은 이동환경에서 잠금 작업을 하거나 충돌 중인 데이타를 찾아내는 부담(overhead)이 지나치게 크기 때문에 적합하지 않다[2,8,9]. 클라이언트에서 서버로 상향 링크가 존재한다고 해도, 이 채널은 일반적으로 통신 용량이 작고 많은 수의 클라이언트로부터 서버로 요청이 쇄도할 수 있다는 잠재적인 문제 때문에 사용을 피하는 것이 바람직하다. 따라서 가능한 한 서버와의 접촉이 없이 읽기 전용 트랜잭션의 일관성 유지가 가능해야 할 것이다. 읽기 전용 트랜잭션의 일관성유지를 위한 몇몇 연구가 수행되었으나[2,3,4,5,9], 재전송의 경우에는 클라이언트에서 재전송 여부를 알아차리지 못하고 트랜잭션을 처리할 수 있으며[2,3,4], 여러 버전의 데이타를 전송하는 방식[5]이나 제어행렬을 사용하는 방식[9]은 각각 방송데이타의 길이가 지나치게 커지거나 제어정보를 전송하기 위한 상향 트래픽의 증가로 비효율적이다.

따라서 본 논문에서는 서버에서 데이타를 방송할 때 클라이언트가 일관성 있는 데이타를 액세스하도록 하는 새로운 알고리즘인 SCDSC(Serialization Checking with DirtySet on Commit)를 제안한다. SCDSC는 클라이언트가 필요로 하는 모든 데이타 아이템이 방송 데이타의 논리적인 단위인 bcast 안에 적어도 한번 삽입되며, bcast가 한번 방송되는 기간을 방송주기로 하는 환경에서 수행된다. SCDSC 기법은 bcast에 지난 일정 주기(d>0) 동안 갱신된 데이타 집합인 변경집합(DirtySet)을 추가시켜 방송한다. 이동 클라이언트가 제기한 읽기 전용 트랜잭션은 자신이 필요로 하는 데이타를 모두 읽은 후 커미트(commit)를 수행하는데 이때 DirtySet 내에 자신이 이미 읽은 데이타가 있는지 검사하여 일관성 및 현재성을 유지한다. 이 논문에서 클라이언트는 읽기 연산만을 수행하고 갱신은 모두 서버에서만 발생하며, DirtySet은 슬라이딩 윈도우 방식으로 방송주기가 바뀔 때마다 갱신되어 서버에서 유지된다고 가정한다. 또한, d의 값은 이동 클라이언트에서 발생하는 하나의 트랜잭션이 처리되는 최대 주기에 의해 결정된다.

본 논문의 나머지 구성은 다음과 같다. 2절에서는 방송 데이타에 대한 정확성의 기준으로 일관성과 현재성을 정의하고 이동 환경에서의 일관성 있는 데이타 방송 기법에 관해 기존의 연구를 살펴보고, 3절에서는 SCDSC 기법이 적용되는 이동 데이타베이스 모델을 정의한다. 4절에서 일관성 있는 방송 데이타를 유지하도록 하는 SCDSC 기법을 설명하고 5절에서 일관성 및 현재성 관점에서 SCDSC 기법을 조망하고, 제안한 알고리즘의 성능을 측정하고 결과를 논의하며 6절에서 결론을 짓기로 한다.

2. 관련 연구

이 절에서는 동시성 제어의 정확성 기준으로 일관성

과 현재성에 대해 설명하고 방송기법에서의 동시성 제어에 관한 연구를 소개한다.

2.1 일관성(consistency)

이동 클라이언트의 트랜잭션은 읽기 연산으로 구성된다. 따라서 이동 클라이언트의 트랜잭션이 일관성 있는 데이타를 읽으면 그 트랜잭션은 일관성이 있다고 할 수 있다. 각각의 서버 트랜잭션이 데이타베이스 일관성을 유지하며 수행된다고 가정한다면, 읽기 전용 트랜잭션에 의해 읽혀진 값은 일관성 있는 데이타베이스 상태의 부분집합이어야 한다. 따라서 많은 트랜잭션을 직렬화 될 수 있는(serializable) 실행으로 생성된 상태는 일관성 있는 데이타베이스 상태를 생성한다. 직렬성(serializability)은 기존의 데이타베이스 시스템에서 흔히 사용되고 데이타베이스 커뮤니티에서 폭넓게 수용되므로 데이타베이스 일관성의 기준으로 채택이 가능하다[8].

2.2 현재성(currency)

이동 컴퓨팅 환경에서 많은 데이타 아이템은 특정 도로의 현재 교통 조건, 특정 도시의 현재 기상 상태 및 특정 주식의 최신 시세와 같은 실시간 정보를 반영해야 하기 때문에, 그 값은 굉장히 동적이고 민감하다. 데이타베이스 서버에서 이동 클라이언트로 데이타 값이 방송되는 동안, 갱신 트랜잭션이 도착하고 새로운 값이 데이타베이스에 반영된다. 그러한 동적인 환경에서 외부 환경의 객체 상태와 데이타베이스의 해당 데이타 아이템 값 간의 엄격한 일관성을 유지하는 것은 어려운 일이다. 이것은 갱신 과정을 완료하는데 걸리는 지연 때문이며 특히, 원격 사이트에서 갱신이 발생했을 때 더욱 그렇다. 예를 들어, 교통 정보 시스템에서, 특정 도로 상에 차량 추돌로 인한 정체가 발생했다고 가정해보자. 이것은 해당 데이타베이스를 유지하는 데이타 공급자에게 전송될 것이다. 갱신 생성 사이트에서 이 트랜잭션을 전송하는데 지연이 없다고 가정하더라도 데이타베이스 서버에서 생기는 자원 및 데이타 경쟁 때문에 지연은 여전히 생길 수 있다.

대부분의 경우 오래된 정보는 거의 쓸모가 없을 것이다. 앞서의 예처럼 이미 도로 상의 정체가 해소되고 났을 때도 여전히 정체 정보가 유효한 것처럼 나타난다면 오히려 방해가 될 것이다. 이것을 예방하려면, 시스템은 가능한 한 갱신 트랜잭션을 빨리 처리하고 가장 최근에 커미트된 최신 정보를 전송해야 한다. 이동 망에서 가장 최근에 갱신된 데이타 아이템 값을 제공하려면 비용이 많이 들기 때문에 이동 클라이언트의 트랜잭션이 필요로 하는 데이타가 일정한 방송 주기 내에 존재하도록 제한하는 것도 해결방안이 될 수 있을 것이다[4].

2.3 데이타 방송을 위한 동시성제어

이동 컴퓨팅에 있어서의 데이타 관리 문제는 최근 몇 년 동안 활발하게 연구되고 있고[1,10,11], 데이타 방송 기법도 그 중의 하나로 이동 컴퓨팅을 지원한다[1-4, 5,10,12]. 데이타 방송이 이동 컴퓨팅에 적합한 이유는 이동 컴퓨팅이 기본적으로 배포 본위(dissemination-based) 시스템으로 내재적으로 비대칭 통신이라는 점 때문이다. 비대칭 통신은 서버에서 클라이언트로의 하향으로 전송되는 데이타의 양이 상향, 즉 백채널(back channel) 전송량보다 훨씬 더 크므로[1], 1-to-n 혹은 m-to-n(n >> m)의 특성의 데이타 방송기술과 부합된다. 또한 케이블 방송이나 위성 방송, 인터넷 상의 멀티캐스트 기술과 같은 대규모의 배포를 가능하게 하는 통신기술의 발전이 방송기술을 뒷받침할 수 있기 때문이다. 이러한 방송 기법은 다음의 3가지 모델, 푸쉬(push) 기반의 1-way 방송, 풀(pull) 기반의 2-way 방송과 푸쉬와 풀을 통합한 혼성(hybrid) 방송으로 나눌 수 있다[13]. 방송전용 모델인 푸쉬 기반의 1-way 기법에서, 서버는 특별히 요청하지 않아도 주기적으로 데이타를 클라이언트로 방송하며, 클라이언트의 트랜잭션은 무선 채널 상의 방송에서 필요한 데이타를 액세스한다[5,10]. 이때 한 주기 동안 방송되는 데이타 단위인 bcast를 [11]에서 정의하고 있는데, 이는 인덱스 세그먼트에 끼인 모든 데이타 세그먼트, 즉 파일의 각 버전을 의미한다.

데이타 방송에 있어서의 동시성제어, 즉 방송되는 데이타에 대한 일관성을 유지하는 문제에 대한 연구도 근래에 활발히 논의되고 있다[2,5]. 이것은 2PL이나 OCC와 같은 기존의 동시성 제어 프로토콜은 수많은 클라이언트를 가지며 비대칭 구조의 대역폭을 갖는 이동 환경에 적합하지 않기 때문이다[5]. 이러한 동시성 제어 연구 중 [9]는 제어 행렬을 데이타와 함께 방송하여 읽기 전용 트랜잭션이 최신 및 일관성 있는 데이타를 읽을 수 있도록 하는 F-Matrix와 R-Matrix 알고리즘을 제안하고 있다. 제어 행렬은 n개의 데이타 아이템으로 된 데이타베이스 파일에 대해 n x n 크기로 구성되며, 행렬의 각 엔트리 c_{ij}는 데이타 아이템 j에 가장 최근에 커미트된 값에 영향을 주면서 데이타 아이템 i에 기록 연산을 수행한 트랜잭션이 커미트된 사이클 번호이다. 이동 클라이언트는 공중에서 데이타 아이템을 읽기 전에 이 행렬을 사용하여 일관성 검사를 한다. 이 방법은 갱신 트랜잭션뿐만 아니라 읽기 전용 트랜잭션에도 적용할 수 있어 높은 일관성 정도를 지원한다. 그러나 서버가 각 클라이언트에서 수행된 트랜잭션의 Read Set 및 Write Set 정보를 읽어야 하는 부담이 커서 클라이언트 수가 늘어나면 적용이 쉽지 않다는 문제가 있다.

이동 컴퓨팅을 위한 네트워크는 상대적으로 낮은 서비스 품질 때문에 데이타 일관성을 보장하는 것이 쉽지 않다. [12,14]에서는 한 가지 해결 방안으로 일관성 요구

사항을 완화하는 방안인 2단계 일관성 모델을 제안하고 있다. 의미상으로 관련된 데이타는 클러스터로 그룹화되며 클러스터 내의 데이타는 서로 일치해야 한다. 그러나 다른 클러스터 간의 데이타 아이템 간에는 어느 정도의 불일치를 허용함으로써 이동 환경에서 빈번한 네트워크 단절 상황에 적절히 대처할 수 있는 모델을 제시하고 있다.

[5]에서는 일관성 없는 데이타 값을 읽는 문제를 해결하기 위한 방안으로 다중버전 데이타 방송 기법과 주기적인 무효화 보고(invalidation report, IR)를 바탕으로 하는 방법을 제안하고 있다. 다중버전 데이타 방송 기법은 높은 동시성을 제공하나 여러 버전의 데이타를 제공함으로써 방송주기가 길어지고 갱신된 데이타 반영이 늦어져 낮은 현재성을 보이는 반면 IR은 높은 현재성과 낮은 동시성을 나타낸다.

[2,3,4]에 소개된 UFO(Update First with Order) 혹은 *OUFO(Ordered Update First with Order)* 알고리즘은 데이타베이스 서버에서 갱신 트랜잭션과 읽기 전용 트랜잭션 간의 데이타 충돌로 인한 비직렬화를 충돌된 데이타의 재방송(re-broadcast)을 통해 유지한다. 이 알고리즘은 방송을 통해 데이타를 액세스하는 이동 클라이언트에서 직렬화 그래프(serialization graph)를 유지하여 순환(cycle) 부분을 발견하면 해당 부분의 노드를 제거하고 재방송 데이타를 액세스하거나[2,3] 이미 읽은 데이타가 재방송되는 경우 그 데이타를 읽은 이후의 읽기 연산을 다시 수행하므로써 일관성을 유지한다[4]. 이 알고리즘은 서버, 클라이언트 및 방송 스케줄에 영향이 적어 효율적이나 데이타 충돌 횟수가 많아지면 재방송되는 데이타가 늘어날 수 있는 단점이 있다. 특히 방송주기와 비슷한 주기로 갱신이 발생하는 데이타가 있는 경우에는 매 주기마다 다시 전송되어야 하고, 데이타 단위로 방송하므로 재방송으로 인해 다른 데이타의 액세스가 계속 지연될 수 있다.

3. 이동 데이타베이스 시스템 모델

본 논문에서 기반으로 하는 이동 컴퓨팅 시스템 모델은 그림 1에서와 같이 방송 서버, 많은 이동 클라이언트와 이동 네트워크로 구성된다[4]. 방송 서버는 낮은 대역폭을 갖는 이동 네트워크의 무선 채널을 통해 이동 클라이언트와 통신한다. 방송 서버는 이동 클라이언트에 방송할 데이타베이스를 유지하고 동적으로 변하는 정보를 외부의 데이타 공급자로부터 공급받는다.

서버는 공급자로부터 받은 데이타를 데이타베이스에 반영하기 위해 갱신 트랜잭션을 수행하며, 갱신 트랜잭션 간의 데이타 충돌은 2PL-HP와 같은 기존의 실시간 동시성 제어 프로토콜을 사용하여 해결한다[2]. 어떤 변경이든 서버에서 일어나고 배포된다. 초기에 데이타가 선택되어 방송되는 한 기간을 방송주기(broadcast cycle)라 하고, 본 시스템에서는 한 주기 동안 방송되는 방송 데이타와 DirtySet을 bcast라고 한다. 모든 데이타는 주기적으로 방송되며, 방송 주기의 길이는 고정되거나 가변적이고 하나의 방송 주기는 중간에 빈 시간 없이 즉시 다음 주기로 바뀐다.

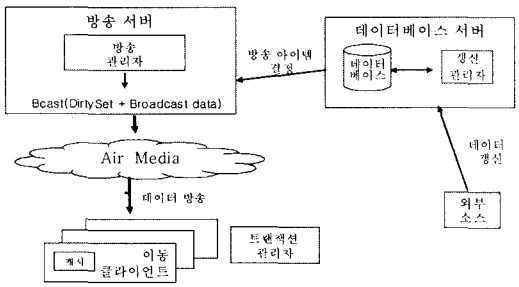

그림 1 이동 데이타베이스 모델

방송되는 데이타 아이템의 값은 방송 사이클 초기의 데이타베이스 상태 즉, 사이클 초기까지 커미트된 트랜잭션에 의해 생성된 값과 일치한다. 따라서 각 방송주기 동안 일반 트랜잭션이 서버에서 처리되고 어떤 아이템은 방송 중에 변경되더라도, 공중의 데이타 값은 변하지 않는다. 방송주기 초반에는 이전 주기에서 변경된 제어 정보와 데이타 아이템 값으로 bcast를 형성한다. 클라이언트는 읽기 전용 방식으로 방송으로부터 데이타를 액세스한다. 클라이언트는 배터리 소모를 가능한 한 줄여야 하기 때문에 원하는 특정 아이템을 읽도록 맞춰져야 한다. 그러기 위해 방송되는 데이타에는 클라이언트가 채널에 관심 있는 데이타가 나타났는지 알 수 있도록 인덱스 정보를 제공한다. bcast 내에서 논리적으로 가장 작은 단위는 버켓(bucket)이다. 버켓은 디스크의 블록과 유사하다. 각 버켓에는 유용한 정보가 담겨있는 헤더가 있다. 버켓 헤더의 정확한 내용은 방송 체계에 따라 다르지만 본 시스템에서는 다음 버켓의 시작 위치까지의 오프셋(offset)과 현재 버켓의 오프셋으로 bcast에서 버켓의 위치를 알 수 있도록 한다. 데이타 아이템은 데이타베이스 레코드와 일치한다. 사용자는 그 레코드의 속성 값 하나 즉, 탐색키로 데이타를 액세스한다고 가정한다. 각 버켓은 몇 개의 데이타 아이템을 포함한다.

한편 이동 클라이언트에는 가장 최근에 갱신된 데이타 정보를 갖는 DirtySet과 최근에 액세스된 데이타 아이템의 일부를 유지하기 위한 캐시가 있다. 캐시에 보관된 데이타 아이템은 버전을 가지며 방송주기가 바뀔 때마다 버전이 증가되며 접속단절 상태에서 깨어날 때나 서버에서 버전을 리셋할 때는 헤더정보를 통해 액세스

하고 처리한다고 가정한다. 캐시의 크기는 이동 클라이언트에 있는 주기억장치의 작은 크기 때문에 제한되기 때문에 모든 데이타 아이템을 보유하지 못한다. 따라서 캐시에 존재하지 않는 데이타에 대한 요청은 방송되기를 기다려서 공중에서 얻어야 한다.

이동 클라이언트에서 수행하는 읽기 트랜잭션 T의 범위, life-span(T)은 데이타를 요구하는 트랜잭션 T가 도착해서 완료될 때까지 걸리는 방송주기로 정의한다. 이동 클라이언트의 트랜잭션은 life-span(T) 동안에 제공되는 몇 개의 bcast로부터 데이타 아이템을 액세스한다. 클라이언트 트랜잭션은 읽기 연산을 모두 마친 뒤 커미트를 시도하며, 몇 개의 다른 주기에서 읽은 데이타에 대한 일관성 검사를 방송 중인 제어정보를 이용해 수행한다.

4. SCDSC 기법

이 절에서는 클라이언트에 일관성과 현재성을 유지하면서 데이타를 방송하는 SCDSC(Serialization Checking with DirtySet on Commit) 기법에 대하여 설명한다.

데이타 방송 시 가장 큰 문제는 데이타 아이템이 방송되고 있는 동안 서버에서 데이타 갱신 트랜잭션이 커미트되어 데이타베이스 상태가 변할 수 있다는 것이다. SCDSC는 클라이언트가 몇 개의 읽기 연산으로 구성된 읽기 트랜잭션을 커미트할 때 데이타와 함께 방송되는 DirtySet을 점검하여 이미 읽은 데이타 중 변경된 데이타가 있으면 변경 데이타를 다시 읽은 후 커미트하여 일관성을 유지한다. DirtySet은 윈도우 크기에 해당하는 일정 주기 동안에 갱신된 데이타 아이템의 인덱스, 상대적 버전과 값으로 구성된다. 다음에서 각각 SCDSC를 위해 서버에서 수행하는 기능, 클라이언트에서 수행하는 기능 및 DirtySet을 구성하는 제어정보가 어떤 내용인지 순서대로 설명하기로 한다.

4.1 서버 수행 기능

서버는 서버 내에서 수행되는 갱신 트랜잭션과 많은 클라이언트에 주기적으로 데이타를 배포하기 위한 방송을 한다. DirtySet은 슬라이딩 윈도우 방식으로 갱신된다. 즉, 바로 이전 주기에 갱신된 데이타의 인덱스, 상대적 버전과 데이타 값이 DirtySet에 추가되고, 가장 이전에 방송주기에 추가되었던 데이타는 버려진다. 또 슬라이딩 윈도우 크기 내에서 다시 갱신된 데이타는 버전과 변경된 값으로 재 기록된다. 그림 2는 서버에서 수행하는 SCDSC에 사용되는 클래스의 일부를 UML로 표현하고 알고리즘을 작성한 것이다.

표 1에 정의된 update DirtySet 루틴은 방송주기 초반에 수행되며 DirtySet에 속하는 모든 데이타의 버전을 갱신한다. 슬라이딩 윈도우 크기를 WindowSize라고 한다면 주기가 바뀜에 따라 버전은 1씩 증가하며 버전 값이 WindowSize인 데이타는 DirtySet에서 제거된다. 또 이전 방송주기 동안 변경된 데이타 아이템은 DirtySet에 반영되는데 이미 DirtySet에 포함된 데이타 아이템은 버전을 최근에 변경된 의미를 갖는 1로 바꾸고 변경된 데이타 값으로 교체한다. 방송되는 데이타 아이템은 일정한 인덱스를 갖는다고 가정하므로 인덱스는 변경하지 않는다. DirtySet에 포함되어있지 않은 데이타 아이템인 경우 DirtySet에 인덱스, 버전과 데이타 값을 추가한다.

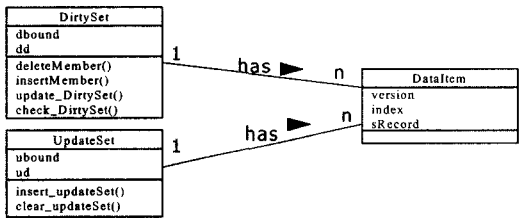

그림 2 서버 내의 DirtySet 관련 클래스

표 1 DirtySet 갱신 알고리즘

```
Algorithm update DirtySet()
  begin
    for each data item di in DirtySet DS
       if (di.version < WindowSize) then
          increment di.version
       else
          delete di  from DS
    for each data item dj in UpdateSet US
       if (dj.version is in DS)
          set dj.version to 1
       else
          insert dj into DS
          set dj.version to 1
  end
```

4.2 클라이언트 수행 기능

클라이언트는 읽기 전용 방식으로 방송 중인 데이타를 액세스한다. 클라이언트가 계속 방송에 귀를 기울여야 할 필요는 없고 특정 아이템을 읽도록 맞춰져야 한다. 클라이언트는 방송 체계에 대해 미리 알고 있어 채널에 원하는 데이타가 나타났는지 결정할 수 있다. 아직 수행할 연산이 남아 있어 커미트하지 못한 활성 트랜잭션인 경우 우선 캐시를 검색하여 원하는 데이타가 있으면 읽고 아니면 공중에서 방송되는 bcast에 있는 해당 데이타 아이템을 읽기 위해 기다린다. 모든 읽기 연산을 수행하여 커미트를 원하는 트랜잭션은 캐시에 포함된

DirtySet 부분에서 커미트할 트랜잭션에서 이미 읽은 데이타가 포함되어 있는지 확인한다. DirtySet에 이미 읽은 데이타가 포함되어 있고 그 버전이 상위라면 그 데이타는 읽은 후 갱신되었다는 것을 의미하므로 이 방송주기에서 다시 읽은 후 커미트한다. 트랜잭션의 마감시간이 지나면 트랜잭션은 취소되며, 트랜잭션의 마감시간은 요청 데이타 중 가장 짧은 마감시간으로 결정된다고 가정한다. 클라이언트는 전체 트랜잭션에 대한 직렬화 그래프를 지속적으로 유지하고 관리하는 대신 각 트랜잭션에 대해 이미 읽은 데이타 아이템의 집합을 상대적 버전 값과 함께 유지하면 된다. 그림 3은 이러한 관계를 나타내는 클래스의 일부이고 표 2는 트랜잭션 커미트를 결정하는 알고리즘을 표현한다.

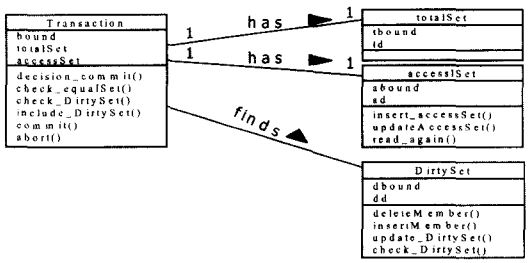

그림 3 클라이언트 내의 트랜잭션 관련 클래스

표 2 트랜잭션 커미트 결정 알고리즘

```
Algorithm decisionCommit(t)
  begin
    while(true)
      for transaction t
        if(current time > deadline(t)) return false
        for each dataItem di in transaction t
          if(di is in DirtySet DS  and di.version >
          DS::dj.version)
                insert di to Re-readSet RS
        if (RS = ∅) return true
        else readAgain(RS)
      end while
  end
Algorithm updateAcessSet()
  begin
     for each data item di in AccessSet AS
       if (di.version < WindowSize) then
          increment di.version
  end
```

클라이언트는 방송주기 초반에 모든 활성 트랜잭션에 대해 updateAccessSet 루틴을 사용해 읽은 데이타 아이템의 버전 값을 1씩 증가시킨다. 모든 데이타 아이템을 다 읽은 트랜잭션은 커미트 여부를 점검하기 위해 decisionCommit 루틴을 호출한다. 이 루틴에서는 트랜잭션 t의 마감시간이 초과하면 false를 리턴하고, DirtySet 내에 이미 읽은 데이타 아이템이 존재하면 그 데이타 아이템의 버전을 비교하고 이미 읽은 데이타 아이템의 버전이 더 크다면 갱신된 데이타이므로 다시 읽어(readAgain(RS)루틴) 커미트 여부를 결정한다.

4.3 bcast 구조

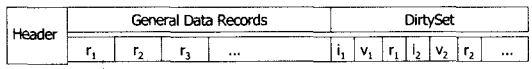

그림 4 bcast 구조

각 방송주기에서 서버는 DirtySet의 버전을 1씩 증가시키고 버전이 WindowSize를 초과하는 데이타 레코드는 버리고 지난 방송주기 동안 변경이 일어난 데이타를 적용한다. 버전은 방송주기의 상대적인 값으로 현재의 방송주기가 ci이고 이전 방송주기를 c_j(j<i, j<= i-d)라고 할 때 i-j가 버전 값에 해당한다. 헤더에는 DirtySet의 시작 지점과 끝 지점의 오프셋이 포함된다. 한 주기 동안 방송되는 데이타는 동일하므로 일반 방송 데이타는 시작 지점에서의 오프셋으로 그 위치를 알 수 있다. 방송되는 데이타 아이템의 크기는 일정하다고 가정하고 데이타베이스 총 데이타 아이템의 개수를 S라고 놓으면, 데이타 아이템의 탐색키는 log256S 바이트로 표현할 수 있다. 방송되는 최대 방송 변경 크기에 해당하는 WindowSize는 life-span을 커버하는 방송주기의 최대값으로 지정할 수 있다. 따라서 WindowSize의 값은 1자리의 작은 정수값으로 선택할 수 있어 버전은 1 바이트로 충분히 표기할 수 있다. 또 탐색키를 제외한 속성을 포함하는 데이타 아이템의 크기를 R로, 평균 갱신비율을 r(0 <= r <=1)이라고 가정하자. 첫 번 방송을 시작해서 d(d > WindowSize)번의 주기가 지나고 난 후 한 주기 동안 방송되는 bcast의 크기는 최대 r * WindowSize * S * (R + log256S +1) + S * R +C (C는 상수, 헤더 크기)이다. 여기서 최대가 의미하는 것은 d 주기 동안 변경된 모든 데이타 아이템의 개수는 r * d * S이지만 실제로 이 기간동안 WindowSize가 지난 데이타는 버려지고 중복 변경 값이 존재하므로 이보다는 적은 개수이기 때문이다.

5. 성능 분석 및 평가

5.1 SCDSC 알고리즘의 일관성 분석

서버에서의 갱신 트랜잭션은 2PL과 같은 기존의 동시성 제어 기법으로 해결된다고 가정하므로 문제가 되는 경우는 어떤 주기에 방송한 데이타가 서버에서 갱신

되는 경우이다. 읽기 트랜잭션은 몇 개의 데이타 아이템, 즉 읽기 연산으로 구성된다. 각 읽기 트랜잭션은 마감시간과 함께 정의된다. 읽기 트랜잭션의 읽기 연산에는 순서가 있기 때문에 한 트랜잭션의 life-span은 몇몇 방송주기가 소요된다고 가정할 수 있다.

다음 예를 보기로 하자. 이동 클라이언트1과 2는 소요되는 주기가 각각 2와 3인 읽기 트랜잭션을 수행한다. 서버에서 방송되는 데이타를 두 개의 클라이언트에서 각기 다른 형태의 순서로 읽지만 두 클라이언트 간에는 읽기-읽기 충돌만이 존재하므로 문제가 발생하지 않는다. 즉, 각 클라이언트에서 읽기 연산의 직렬화만 보장된다면 일관성 문제는 해결된다.

이동 클라이언트1(RT_1): $r_1[x]$, $r_1[y]$, $r_2[z]$, $r_2[w]$, c_2

이동 클라이언트2(RT_2): $r_1[y]$, $r_2[x]$, $r_2[z]$, $r_3[a]$, $r_3[x]$, c_3

서버(UT) : $w_1[x]$, $w_1[y]$, $w_2[y]$, $w_3[w]$

RT_i: Read Transaction at client i

UT: Update Transaction

r_i : read operation at cycle i

w_i : write operation at cycle i

c_i : commit at cycle i

RT_1은 r_1 -> w_1 -> r_2 형태로 순환(cyclic) 그래프가 형성되지만 읽기 연산을 마치고 커미트 시 DirtySet 영역에서 이것을 발견할 수 있고, 변경된 데이타를 다시 읽음으로써 w_1 -> r_3로 순환 부분을 없애고 커미트할 수 있다. 마지막 읽기 연산이 일어나는 방송주기에서 방송 중인 데이타에 갱신이 발생하지만 실제로 방송이 끝나는 말기 부분에 반영되기 때문에 직렬화에는 문제가 되지 않는다. 클라이언트2에서 수행되는 RT_2도 커미트할 때 r_1 -> w_1 -> r_2 -> w_2 -> r_3의 사이클을 발견할 수 있다. 이것은 갱신이 발생한 데이타를 DirtySet에서 다시 읽음으로써 w_1 -> w_2 -> r_3의 비순환(acyclic) 형태로 만들 수 있다.

5.2 SCDSC 알고리즘의 현재성 분석

SCDSC 기법은 방송 중에 변경되는 데이타를 방송주기마다 DirtySet에 추가시킴으로써 커미트 시 데이타 아이템에 최신 데이타를 반영할 수 있도록 한다. 그러나 방송할 파일이 커서 방송주기가 길어진다면 말기 부분에 실제 작업이 발생하는 갱신 트랜잭션의 대기 시간 또한 길어진다. 따라서 오랫동안 갱신 트랜잭션을 블로킹하게 되므로 가장 최신의 데이타를 공급하기는 어려운 단점이 있다. 4.3절에서 한 주기 동안 방송되는 bcast의 최대 크기는 $r * d * S * (R + \log_{256}S + 1) + S * R + C$(C는 상수, 헤더 크기)로 계산되었다. r은 최대값이 1이고 d는 DirtySet의 WindowSize로 가정하며, S, R은 파일 크기에 영향을 주는 아이템으로 S*R은 파일의 크기이다. 그러나 슬라이딩 윈도우 동안 계속 모든 데이타가 갱신된다고 가정한다면, 최신 데이타만 전송되므로 DirtySet의 크기는 $d * S * R + d * S * \log_{256}S + d * S$가 아니라 $S * R + S * \log_{256}S + S$이다. 따라서 bcast의 최대 크기는 $2 * S * R + S * \log_{256}S + S + C$이며 S 값은 S*R에 비해 훨씬 작으므로 크게 2*S*R로 볼 수 있다. 즉, 최악의 경우 파일 크기의 2배이다. 평균적으로 전체 데이타에 대한 갱신 비율은 낮은 값을 가지지만 2배 한도 내에서 방송되는 파일 크기에 비례한다고 할 수 있다.

5.3 시뮬레이션을 통한 성능 실험 및 결과 분석

SCDSC 알고리즘의 성능 척도로 클라이언트 트랜잭션의 마감시간초과율(deadline miss ratio)과 응답시간(response time)을 사용하였다. 마감시간초과율은 마감시간을 넘겨 중단된 트랜잭션의 수를 생성된 전체 트랜잭션 수로 나눈 값의 백분율로 정의한다. 이것은 트랜잭션이 시간 요구사항을 만족시키는 정도를 나타내며, 읽기 트랜잭션에 의해 관찰된 데이타 아이템의 동시성에 대한 중요한 척도이다. 평균응답시간은 클라이언트에서 트랜잭션을 제기한 시간부터 커미트 될 때까지의 시간으로 정의한다. 실험은 Intel Pentium 600MHz CPU를 탑재한 리눅스(Linux) 환경에서 gcc를 사용해 수행하였다.

5.3.1 성능 평가 모델

성능 평가 모델의 주 구성요소는 갱신 트랜잭션 생성기, 데이타베이스 서버 내의 갱신 트랜잭션 관리자와 방송을 담당하는 방송 관리자, 이동 클라이언트의 읽기 트랜잭션 관리자이다. 갱신 트랜잭션 생성기는 외부에서 변경된 데이타를 제기하는 트랜잭션을 포아송(Poisson) 분포에 따라 생성한다. 갱신 트랜잭션 관리자는 데이타베이스에 갱신 트랜잭션을 수행하며, 갱신 트랜잭션 간의 데이타 충돌은 2PL-HP를 사용하여 해결한다고 가정한다. 방송 관리자는 방송주기 초반에 bcast를 방송한다. 각 방송주기는 bcast를 방송하는 시간으로 bcast 크기에 따라 가변적이다.

읽기 트랜잭션 관리자는 읽기 전용 트랜잭션을 생성한다. 각 클라이언트는 한번에 하나의 트랜잭션을 생성하며 포아송 분포로 트랜잭션을 생성한다. 트랜잭션 T는 도착시간+life-span(T)을 마감시간으로 갖는다고 가정한다. 읽기 트랜잭션을 마감시간이 지나서 완료하게 되면 유용성이 훨씬 떨어져 완전히 쓸모없게 되므로 트랜잭션은 마감시간 후에 중단된다. 각 트랜잭션은 일련의 연산으로 구성되며, 각 연산은 몇 개의 데이타 아이템을 액세스할 것을 요구한다. 읽기 트랜잭션은 일단 생성되면, 첫 번째 연산에서 필요한 데이타 아이템을 얻기 위해 방송 사이클에 귀를 기울이고 다음 연산도 마찬가지로 처리한다. 읽기 트랜잭션은 자신의 연산을 모두 처리한 후에 커미트를 수행하며 이때 DirtySet을 액세스

한다.

5.3.2 성능 평가 파라미터

시뮬레이션에서는 데이타베이스 크기에 대해 갱신 정도와 DirtySet의 윈도우 크기에 따른 bcast 크기, 평균 응답시간 및 마감시간초과율 간의 관계를 조사하고 갱신 정도에 따라 기존의 UFO 방식과 비교했을 때 어떤 성능을 나타내는지 살펴본다. UFO 방식은 최근에 제안된 동시성 제어 알고리즘이며, 이전의 알고리즘인 무효화 보고 방식과 다중버전 방식과 성능 비교를 통해 성능 상의 우수성이 입증되었다[3]. 이 알고리즘은 클라이언트의 명시적인 요구에 의한 갱신 데이타의 재방송으로 일관성을 유지하기 때문에 본 알고리즘과 같은 묵시적인 방법과 비교할만하다. 표 3에 주요 성능 파라미터와 기본적인 값을 표시하였다. 본 실험에서 사용하는 시간 단위는 하향 링크 채널 용량을 나타내지 않기 위해 한 페이지를 방송하는데 필요한 시간으로 정의한다. 데이타베이스 크기는 각각 데이타 아이템 500개와 1,000개로 구성된 소규모와 일반 크기를 가정한다. 데이타 아이템의 크기는 서로 다르지만 평균 크기를 10 페이지로 가정한다. 이동 클라이언트의 수는 50으로 가정하며, 갱신 트랜잭션 및 읽기 트랜잭션에서 갱신 및 요청하는 데이타 아이템 수는 각각 1에서 5와 1에서 3까지로 하며, Zipf 분포를 따른다고 가정한다. 이것은 갱신 및 요청 트랜잭션의 대상 데이타의 수가 적은 쪽으로 편중되어 있기 때문이다. 읽기 트랜잭션의 평균 life-span이란 트랜잭션의 마감시간을 의미하는 것으로 정규분포를 따른다고 가정한다. 트랜잭션의 평균도착시간 간격은 일반적으로 많이 사용하는 워크로드(workload)와 비슷하게 설정하였다[3,15].

표 3 성능 파라미터

성능 파라미터	기본 값
데이타베이스 크기	500~1,000 items
데이타 아이템 크기	10 페이지
이동 클라이언트의 수	50
갱신 중복 비율	0.2
갱신 트랜잭션의 평균 연산 수	1 to 5
읽기 트랜잭션의 평균 연산 수	1 to 3
읽기 트랜잭션의 평균 life-span	50,100,150 단위시간
갱신 트랜잭션의 평균 도착시간 간격	1 to 17 단위시간
읽기 트랜잭션의 평균 도착시간 간격	20 단위시간
WindowSize	4 사이클

5.3.3 성능 평가 실험 결과분석

그림 5,6,7은 갱신 트랜잭션 발생 비율에 따라 bcast의 크기, 평균응답시간 및 마감시간초과율이 SCDSC와 UFO 알고리즘을 사용했을 때 어떤 영향을 받는지에 대해 나타낸다. 그림 5를 보면 SCDSC 기법에서는 bcast 크기가 데이타베이스 크기에 상관없이 갱신 트랜잭션의 평균 도착시간이 2인 지점에서 급격히 작아져 갱신 트랜잭션 빈도가 낮아짐에 따라 조금씩 감소함을 보이는 반면에 UFO 기법에서는 오히려 다소 증가함을 볼 수 있다. 이것은 UFO 기법에서 반복적으로 갱신되는 데이타를 방송주기마다 추가해야 하기 때문인 것으로 판단된다. 그림 6에서 UFO 방식은 갱신 트랜잭션의 도착비율이 낮아져도 평균응답시간이 개선되지 않으나 SCDSC 기법은 응답시간이 점점 낮아짐을 보여준다. 그림 7의 마감시간초과율은 데이타베이스의 크기가 작은 경우에 UFO나 SCDSC 기법 모두 별다른 영향을 받지 않고 일정하게 낮은 값을 보이나 데이타베이스 크기가 큰 경우 UFO는 트랜잭션 갱신 비율이 커지면 증가하는 것을 볼 수 있다. 이것은 방송주기 시간이 일정 한도를 넘게 되어 취소되는 트랜잭션이 많아지기 때문이다.

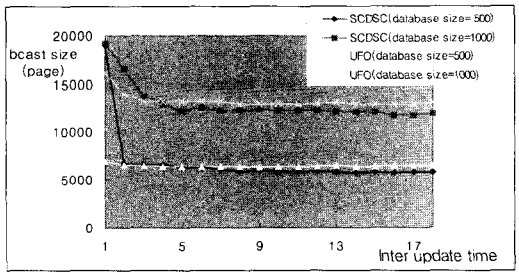

그림 5 bcast size vs. Inter update time

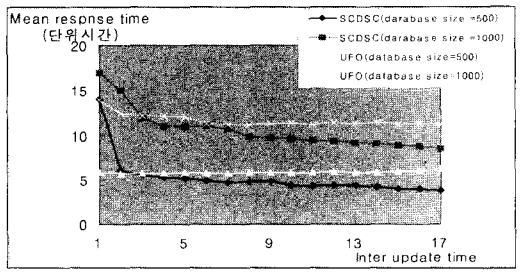

그림 6 Mean response time vs. Inter update time

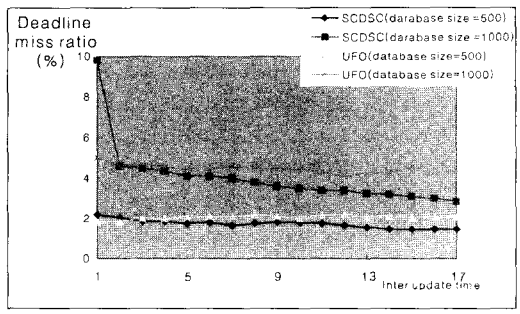

그림 7 Deadline miss ratio vs. Inter update time

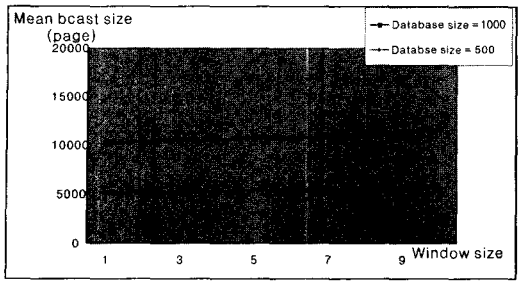

그림 8 bcast size vs. Window size

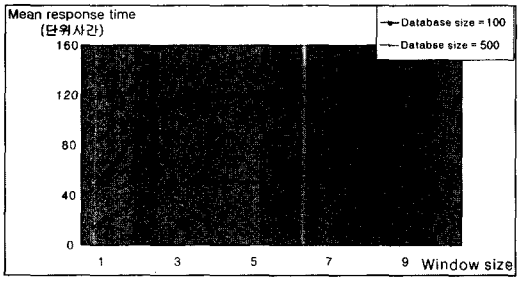

그림 9 Mean response time vs. Window size

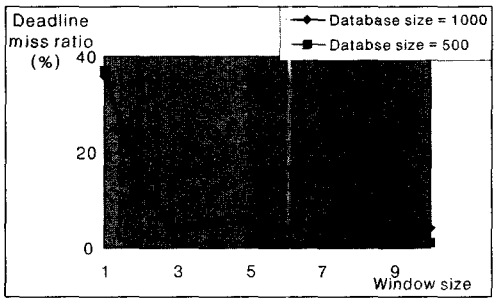

그림 10 Deadline miss ratio vs. Window size

그림 8,9,10은 윈도우 크기가 증가함에 따라 bcast의 크기, 평균응답시간 및 마감시간초과율이 어떤 영향을 받는지 나타낸다. 그림 8에서 bcast의 크기는 윈도우 크기가 커짐에 따라 완만하게 증가한다. 이것은 bcast 크기가 최악의 경우 데이타 파일의 2배까지 될 수 있으나 갱신 데이타는 중복되는 경우가 많으므로 실제로는 크게 늘어나지 않기 때문이다. 그림 9의 윈도우 크기가 4인 지점까지 평균응답시간이 크게 증가하다 이 지점을 지나면 서서히 증가하는 것으로 나타난다. 이것은 윈도우 크기에 따라 bcast가 영향을 받고 주기가 길어져 응답시간이 증가하지만 트랜잭션의 평균수명(life-time)을 지나게 되면 대부분의 트랜잭션이 완료되므로 평균응답시간도 크게 영향을 받지 않는 것으로 분석된다. 그림 10은 마감시간초과율이 윈도우 크기가 3인 지점까지는 크게 감소하다 이 지점이 지나면 서서히 증가하는 것으로 바뀌는데 이것은 읽기 트랜잭션의 평균수명을 초과하는 갱신 데이타는 마감시간초과율을 줄이는데 도움이 되지 않기 때문이다. 결과적으로 데이타베이스 크기, 트랜잭션의 평균수명, 갱신 트랜잭션의 발생 비율에 따라 DirtySet의 윈도우 크기를 결정할 필요가 있고 데이타베이스 크기가 상대적으로 작거나 데이타베이스 크기가 크다면 응답시간이 마감시간초과율보다 중요한 성능척도로 사용될 때 본 알고리즘이 적합한 것으로 볼 수 있다. 또한 UFO와 비교를 통해 클라이언트에서 서버로의 상향링크 대역폭이 절대적으로 부족하고 클라이언트의 전력 소모를 줄일 필요가 있을 때 효율적임을 알 수 있다. 갱신비율이 지나치게 높아지면 UFO에 비해 성능이 떨어지나 본 실험에서는 상향링크 대역폭을 고려하지 않았으므로 실제로 UFO의 성능이 이 실험의 결과보다 훨씬 낮아질 것으로 예상된다.

6. 결 론

휴대폰이나 팜탑과 같은 무선기기를 통한 이동 컴퓨팅 환경이 대중화되면서 데이타베이스 서버에서 많은 무선기기에 내재한 클라이언트로 일관성 있는 데이타를 제공하는 것이 중요한 이슈로 자리 잡았으며, 데이타 방송 기법이 이러한 환경에 적합하다는 사실이 여러 연구에서 밝혀졌다[1-3,14]. 갱신 트랜잭션을 방송과 병행수행하면서 일관성 있는 데이타를 방송하려면 동시성제어 문제가 해결되어야 하며 기존 동시성 기법은 많은 클라이언트와 비대칭 무선 채널 때문에 한계가 있다. 따라서 이러한 환경에 적합한 알고리즘으로 SCDSC 기법을 제안하고 그 성능을 측정하였다. 제안한 알고리즘은 대체로 작은 크기의 데이타베이스 방송에 적합하며, 규모가 큰 데이타베이스의 경우에는 응답시간이 마감시간초과율보다 더 중요한 성능 척도인 환경에 적합하다고 할 수 있다.

참 고 문 헌

[1] S. Acharya, M. Franklin, S. Zdonik, "Balancing Push and Pull for Data Broadcast," in Proceedings of ACM SIGMOD, Tucson, Arizona, May 1997.

[2] K.Y. Lam, E. Chan, and M.W. Au, "Broadcast of Consistent Data to Read-Only Transactions from Mobile Clients," in Proceedings of 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, Louisiana, USA, 1999.

[3] M.W. Au,E. Chan, and K.Y. Lam, "Concurrency Control For Mobile System With Data Broadcast," Journal of Interconnection Networks 2001.

[4] K.Y. Lam, E. Chan, H.W. Leong, and M.W. Au, "Broadcasting Consistent Data to Mobile Clients

with Local Cache," in Proceedings of 2000 International Conference on Management of Data (COMAD 2000), India, December 2000.
[5] E. Pitoura, "Supporting Read-Only Transactions in Wireless Broadcasting," in Proceedings of the DEXA '98 Workshop on Mobility in Databases and Distributed Systems, August 1998.
[6] O. Ulusoy, "Real-Time Data Management for Mobile Computing," International Workshop on Issues and Applications of Database Technology (IADT'98), Berlin, Germany, July 1998.
[7] Ersan Kayan, O. Ulusoy, "An Evaluation of Real-Time Transaction Management Issues in Mobile Database Systems," The Computer Journal(Special Issue on Mobile Computing), vol.42, no.6, 1999.
[8] Bernstein, P.A., Hadzilacos, V. Goodman, N., Concurrency Control and Recovery in Database System, Addison-Wesley Publishing Company, 1987.
[9] J. Shanmugasundaram, A. Nithrakashyap, R. Sivasankaran, and K. Ramamritham, "Efficient Concurrency Control for Broadcast Environments," ACM SIGMOD International Conference on Management of Data, 1999.
[10] S. Acharya, R. Alonso, M. Franklin and S. Zdonik, "Broadcast Disks: Data Management for Asymmetric Communication Environments," in Proceedings of ACM SIGMOD, 1995.
[11] T. Imielinski, et al. "Data on Air: Organization and Access," IEEE TKDE, 9(3): 353-372, 1997.
[12] E. Pitoura and B. Bhargava, "Maintaining Consistency of Data in Mobile Distributed Environment" in Proceedings of the 15th International Conference on Distributed Computing Systems, pp. 404-413, 1995.
[13] K. Stathatos, N. Roussopoulosand J. S. Baras, "Adaptive Data Broadcasting Using Air-Cache," Proceedings of the 1st International Workshop on Satellite-based Information Service, Rye, NY, 1996.
[14] E. Pitoura and B. Bhargava, "Data Consistency in Intermittently Connected Distributed Systems," in IEEE Transaction on Knowledge and Data Engineering, 11(6), pp 896-915, Nov/Dec 1999.
[15] J. Fernandez and K. Ramamritham, "Adaptive Dissemination of Data in Time-Critical Asymmetric Communication Environments," Proceedings of the 11th Euromicro Conference on Real-Time Systems, Sep. 1998.

윤 혜 숙

1986년 서울대학교 계산통계학과 학사 1988년 서울대학교 계산통계학과 석사 1988년~1995년 한국통신 연구원. 1999년~현재 충남대학교 전기정보통신공학부 박사과정. 2002년 8월~충남대학교 전기정보통신공학부 BK 전임교수. 관심분야는 이동데이타베이스시스템, 실시간 데이타베이스 시스템

김 영 국

1985년 서울대학교 계산통계학과 학사 1987년 서울대학교 계산통계학과 석사 1995년 버지니아대학교 컴퓨터과학과 박사. 1995년 VTT(Technical Research Centre of Finland) 방문연구원. 1995년 SINTEF Telecom & Informetics, Norway 방문연구원. 2002년 8월~2003년 7월 UC Davis 객원교수. 1996년~현재 충남대학교 전기정보통신공학부 부교수. 관심분야는 실시간데이타베이스시스템, 전자상거래시스템, 이동데이타베이스시스템