韓國IT서비스學會誌
第8卷 第1號
2009年 3月

이동 데이터 서비스에서 동적인 데이터 요청패턴을 지원하는 효율적인 데이터 방송 시스템*

신 동 천**

An Efficient Data Broadcast System Supporting Dynamic Data Request Patterns in Mobile Data Services*

DongCheon Shin**

■ Abstract ■

In wireless environments, it is very effective to introduce a data broadcast system for providing a number of clients with mobile data services. In particular, an efficient broadcast scheduling and a caching strategy are very important for performance of data broadcast systems in wireless environments which have such inherent restrictions as low bandwidth, frequent disconnections, and short battery life. In this paper, we present a data broadcast scheduling strategy using bit vectors to cope with dynamic request patterns of clients. In addition, we also propose an efficient caching strategy, 2FWT, that takes characteristics of the broadcast scheduling strategy into account. Finally, we evaluate performance of the data broadcast system. According to the results, the proposed system generally shows better performance than others.

Keyword : Data Broadcast, Caching Strategy, Mobile Data Service

논문투고일 : 2008년 07월 15일 논문수정완료일 : 2008년 09월 08일 논문게재확정일 : 2008년 09월 18일
* 이 논문은 2007년도 중앙대학교 학술연구비 지원에 의한 것임.
** 중앙대학교 정보시스템학과 교수

1. 서 론

최근 들어, 무선통신 기술의 발달로 인하여 이동 컴퓨팅이라는 새로운 컴퓨팅 패러다임에 대한 관심이 고조되고 있다. 이동 컴퓨팅 환경의 특징은 크게 무선과 이동성으로 나누어 볼 수 있다[4, 9]. 기존의 유선망과 비교하여 무선망은 상대적으로 낮은 대역폭(bandwidth), 잦은 단절 등 여러 가지 환경적인 제약요인이 있다. 이러한 이동 컴퓨팅 환경의 특징 때문에 기존의 컴퓨팅 환경에서 제안된 여러 가지 기법이나 알고리즘들은 적용될 수 없거나 낮은 성능을 보이게 된다. 예를 들어, 캐시 관리, 질의처리, 트랜잭션 처리 문제에 무선과 이동성이라는 요인을 고려한 기법들이 효율적이게 된다[11, 12].

제한된 대역폭에서 다수의 클라이언트에게 데이터를 전송하는 방법으로 데이터 방송(data broadcast) 기법이 관심을 끌어 왔다[3, 16]. 데이터 방송 시스템은 방송되는 데이터 항목(페이지)의 선정과 관련하여 크게 3가지 형태로 구분 할 수 있다. 서버가 클라이언트의 데이터에 대한 방송요청 없이 자체 방송 알고리즘에 따라 필요한 데이터를 주기적으로 혹은 비 주기적으로 방송하는 push 기반의 방송 형태가 있다[7]. 반면에 필요한 데이터를 클라이언트가 서버에 요청하여 서버가 데이터를 방송해 주는 pull 기반의 형태가 있다[1]. 그리고 push와 pull 기반을 혼합하여 방송하는 혼합(hybrid) 기반의 형태가 있다[2, 7].

Push 기반의 방송에서는 클라이언트가 필요로 하는 데이터를 방송하는 것이 시스템 성능에 절대적인 영향을 미치게 된다. 그러나 클라이언트의 데이터 액세스 요청에 대한 패턴을 정확하게 파악할 수 없으므로 클라이언트의 액세스 요청 변화 패턴이 변하는 경우에 능동적으로 대처할 수 없는 근본적인 문제점을 갖게 된다. 한편 pull 기반의 방송 형태에서는 클라이언트의 데이터 방송 요청 정보를 기반으로 서버가 데이터를 방송하므로 이러한 문제는 완화된다고 볼 수 있지만 제한된 대역폭의 효율적인 활용이라는 측면에서 문제가 될 수 있다. 특히 업스트림 채널의 낮은 대역폭으로 인하여 클라이언트들의 방송요청이 제한되거나 서버의 포화 현상을 야기할 수 있다. 이러한 문제들의 대안으로 생각할 수 있는 방송 형태가 혼합 형태이다. 혼합 형태의 방송에서는 방송 알고리즘에 따라 생성된 방송 스케줄의 데이터와 클라이언트에서 방송 요청된 데이터를 함께 방송하는 것이다.

지금까지 제안된 대부분의 push 기반 및 혼합 형태의 방송 알고리즘들은 클라이언트의 데이터 액세스 요청에 대한 변화를 동적으로 반영하여 방송 프로그램을 생성하지 못하였다. [7]에서는 비트벡터를 이용하여 클라이언트의 데이터 액세스 요청 변화를 동적으로 반영하여 방송하는 혼합형태 기반의 방송 알고리즘을 제안하였다. 그러나 최근에 액세스 요청된 데이터와 과거에 액세스 요청된 데이터를 차별화하지 않아 과거에 클라이언트들이 자주 액세스 요청했던 데이터들이 방송 스케줄에 여전히 포함되는 비효율성을 초래할 수 있게 된다. 즉, 최근에 인기가 증가하게 되어 액세스 요청이 증가하고 있는 데이터에 대한 클라이언트의 액세스 요청에 효율적으로 대처할 수 없게 된다. 따라서, 클라이언트의 데이터 액세스 요청 패턴 변화에 능동적으로 대처할 수 있도록 방송 알고리즘에 대한 개선이 필요하다.

한편, 데이터 방송 환경에서는 낮은 대역폭으로 인하여 클라이언트에 캐시를 두어 필요한 데이터가 방송되기를 기다리지 않고 캐시를 액세스하여 효율성을 높일 수 있다. 캐시에 있는 데이터는 서버에 있는 데이터의 사본으로 간주될 수 있으므로 일관성 유지 문제가 대두되고 이러한 일관성 문제를 해결하고자 하는 여러 노력이 있어 왔다[5, 8, 10, 14]. 캐시를 고려하는 방법에서 해결해야 될 또 다른 중요한 문제는 캐싱 전략, 즉 캐시내의 데이터 페이지를 새로운 페이지와 대체시키는 대체 전략이다[3, 15, 17].

전통적인 시스템 환경을 위한 여러 가지 대체 전략들이 제안되었다[6]. 그러나, 이러한 대체 전

략들은 이동 컴퓨팅 환경의 특징을 고려하지 않은 전략들이기 때문에 좋은 성능을 보이기 어렵다. 따라서, 이동 컴퓨팅 시스템에서 데이터 방송 환경을 고려한 많은 대체 전략들의 대부분은 캐시의 히트율을 높이는 것뿐만 아니라 캐시 미스가 발생하는 경우 서버로부터 필요로 하는 데이터가 방송될 때까지 기다려야 하므로 미스에 따른 비용까지 고려해야 한다는 관점에서 제안되었다.

데이터 방송 시스템에서 궁극적으로 캐시 전략은 방송 알고리즘과 긴밀한 상호 의존성을 갖는다. 예를 들어, 자주 방송되는 데이터는 상대적으로 드물게 방송되는 데이터보다 캐시에 존재할 가치가 떨어지며 캐시에 존재하지 않는 데이터는 존재하는 데이터보다 상대적으로 자주 방송해야 대기시간을 줄여서 미스 비용을 최소화할 수 있다. [15]을 제외하고 지금까지 제안된 캐시 전략의 대부분은 방송 알고리즘과 독립적으로 제안되었거나 방송 알고리즘의 기본 원칙을 효율적으로 반영하지 못하였다고 할 수 있다.

본 논문은 [15]에서 개선한 방송 스케줄링 전략을 기반으로 캐싱 전략을 수정하여 전체적으로 데이터 방송 시스템의 성능 변화를 고찰하는 데 그 목적이 있다. 캐싱 전략을 수정하기 위하여 데이터의 인기도를 반영하는 클라이언트의 데이터 액세스 요청 빈도 뿐만 아니라 캐시내에 존재 필요성을 반영해 주는 데이터 방송 요청 빈도를 고려한다. 또한 클라이언트의 데이터 액세스 요청의 최신성을 반영하기 위하여 각 고려요소에 방송 버전 번호를 도입한다. 한편, [15]에서는 캐시 존재기간과 방송 횟수를 고려하고 있으나 이는 중복적인 고려 요소라 할 수 있다. 캐시에 오래 존재하면 비트 벡터를 이용하는 방송 스케줄링 특성에 따라 방송 가능성이 점차로 감소하게 되므로 캐시 미스가 발생하는 경우 클라이언트의 대기시간이 길어지게 된다. 또한, 방송 횟수가 많다는 것은 대기시간이 짧아질 가능성이 높음을 의미한다. 이런 측면에서 볼 때, 캐시 존속기간과 방송 횟수 고려는 대기시간과 연관되는 중복된 고려요소라 할 수 있으므로 본 논문에서는 대기시간을 고려요소로 단일화 한다. 끝으로, 이러한 고려 요소의 변화가 데이터 방송 시스템 성능 변화에 미치는 영향을 알아보기 위하여 시뮬레이션을 통해 성능을 평가한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 데이터 방송 스케줄링 전략의 개선에 대해 기술하고 제 3장에서는 새로운 고려 요소들을 반영한 캐싱 전략을 제시한다. 제 4장에서는 성능 평가에 대해 논의한다. 끝으로, 제 5장에서는 결론을 맺는다.

2. 데이터 방송 스케줄링 전략

[15]에서는 클라이언트가 최근에 액세스 요청한 데이터일수록 더 많이 방송되도록 하여 최근에 증가하고 있는 클라이언트의 데이터 액세스 요청 변화에 효율적으로 대처할 수 있도록 [7]에서 제안한 스케줄링 전략을 개선하였다. 이 절에서는 [15]에서 제시한 방송 스케줄링 전략을 기술한다.

논의하는 데이터 방송 시스템은 혼합형태의 시스템이다. 즉, push 형태로 방송되는 데이터와 pull 형태로 방송되는 데이터 사이에 대역폭을 효율적으로 할당하여 두 형태의 방송 데이터를 모두 지원한다. 따라서, 클라이언트의 데이터 방송 요청은 FIFO 형식으로 큐에 저장되어 할당된 대역폭 범위 내에서 일반 방송 프로그램의 데이터와 번갈아 가며 방송된다. 큐가 만원일 때 클라이언트의 방송 요청은 거절되며 이미 큐에 있는 데이터에 대한 방송 요청은 앞서 요청된 데이터 방송으로 만족될 수 있으므로 큐에 저장되지 않는다.

각 방송 주기 초에 방송 스케줄 즉, 데이터 식별자와 방송 주기 버전번호를 먼저 방송한다. 각 클라이언트는 수신한 방송 스케줄 정보를 토대로 필요한 데이터가 캐시에 없는 경우 이번 방송 주기에 방송되면 기다리고 그렇지 않으면 서버에게 해당 데이터를 요청한다.

방송된 데이터에 대한 클라이언트의 액세스 정보를 얻기 위해 각 클라이언트는 비트 벡터를 유

지한다. 비트 벡터의 각 비트는 현재 방송 프로그램의 각 데이터 페이지를 나타낸다. 새로운 방송 프로그램을 수신하면 클라이언트는 버전 번호를 갱신하고 비트 벡터를 모두 '0'으로 클리어 한다. 액세스 요청한 데이터가 방송되는 데이터에 있으면(air hit) 해당 데이터 비트는 '1'로 변경된다. 캐시에 있는 경우나 방송요청하여 방송된 데이터인 경우에는 비트 벡터에 영향을 주지 않는다. 즉, 데이터 비트가 '1'임은 원하는 데이터를 방송했음을 의미하며 반대로 불필요하게 방송된 데이터의 해당 비트는 '0'임을 의미한다. 필요한 데이터가 캐시나 방송 프로그램에 없는 경우 클라이언트는 서버에게 데이터 방송 요청을 한다. 이때, 비트 벡터도 함께 피기배킹(piggybacking)된다. 즉, 데이터 방송 요청 메시지는 데이터 식별자, 주기 버전번호, 비트 벡터로 구성된다. [그림 1]은 비트벡터를 유지하는 예를 보여준다. 서버가 방송하는 데이터 중에서 a, d, g(에어히트 경우)를 제외한 나머지 데이터(b, c, e, f)는 클라이언트가 원하지 않거나(b, f) 원하는 데이터가 캐시에 존재하므로(c, e) 방송 효과가 없다. 따라서 이러한 방송 효과를 반영하여 비트 벡터는 <1001001>가 된다. 캐시 미스가 된 데이터(j)의 방송 요청 메시지를 서버에게 보낼 때 비트 벡터도 함께 보내져 다음 방송 스케줄링에 고려하게 된다.

서버는 각 데이터에 대해 MFA(most frequently accessed) 값을 유지한다. 서버는 버전번호를 확인하고 데이터가 클라이언트에 의해 방송요청 되었거나 데이터의 해당 비트 벡터가 '1'인 경우 해당 데이터의 MFA 값을 '1' 증가시킨다. MFA 값이 높은 데이터들은 방송 효과가 있었음을 의미하므로 방송 디스크 기법[3]에 따라 방송하게 된다. 이렇게 함으로써 클라이언트의 데이터 액세스 정보를 다음 방송 프로그램 생성에 반영한다.

- 서버 데이터 : a, b, c, d, e, f, g, h, i, j
- 캐시내 데이터 : i, c, e
- 데이터 방송 프로그램 : a, b, c, d, e, f, g

- 액세스되는 데이터 : i, a, c, e, d, g, j

- 캐시 히트 : i, c, e
- 에어 히트 : a, d, g
- 캐시 미스 : j

- 비트 벡터 : <1001001>

[그림 1] 비트 벡터 유지 예

그러나, MFA 값을 단순하게 증가시켜 높은 값을 갖는 데이터들을 방송하게 되면 이전에 클라이언트들에게 인기가 높았지만 현재는 인기가 떨어진 데이터들도 당분간은 높은 값을 유지할 수 있어 방송될 가능성이 높아진다. 특히, 새로이 인기가 올라가는 데이터에 대한 방송이 당분간 지연되거나 방송 빈도수가 저하될 가능성이 있다. 예를 들어, 주식거래, 교통정보, 날씨정보, 주문형 뉴스, 주문형 비디오를 포함하여 향후 보편화될 이동 데이터 서비스에서는 한 시점에 인기가 급속히 올라가다가 차츰 시간이 지나면서 서서히 인기가 내려가는 양상을 보일 수 있다. 이러한 서비스에서는 이동과 무선 환경의 근본적인 제약점을 고려할 때 이러한 데이터 요청 패턴 변화에 조금이라도 능동적이고 효율적으로 대처하는 전략들이 더욱 필요하게 된다. 이러한 문제를 최소화하기 위해 다음과 같은 고찰을 할 수 있다.

비트 벡터에 '0'을 갖는 데이터는 클라이언트의 캐시에 있거나 불필요한 데이터임을 의미한다. 캐시에 있는 데이터는 점차 상대적으로 MFA 값이 낮아지게 되지만 캐시에 있으므로(캐시 대체 전략이 이상적이라면) 방송여부나 방송 빈도수의 변화가 문제되지 않는다. 따라서, 불필요한 데이터라면 MFA 값을 상대적으로 빠르게 낮추어 줄 필요가 있다. 한편, 비트 벡터에 '1'을 갖는 데이터는 향후에 자주 액세스 요청될 가능성이 있는 데이터이므로 방송하거나 혹은 방송 빈도수를 더 늘려 줄 필요가 있다. 그러므로 최근에 액세스 요청되는 데이터가 그렇지 않은 데이터에 비해 상대적으로 자

주 방송되는 것을 가속화하기 위하여 비트 벡터가 '1'인 데이터만 고려하지 않고 '0'인 데이터도 다음과 같이 고려하여 MFA 값을 유지한다.

(R1) : 현재 방송 주기동안 클라이언트들로부터 수신한 각 비트 벡터에서 각 데이터별로 비트가 '1'인 개수의 합에서 '0'인 개수의 합을 뺀 값을 해당 데이터의 MFA값에 더한다.

(R2) : 클라이언트들로부터 방송요청된 데이터에 대해 데이터별로 요청된 횟수의 합을 해당 데이터의 MFA 값에 더한다.

따라서 최근에 air 히트가 많은 데이터(비트가 '1'인 데이터) 즉, 방송 효과가 큰 데이터일수록 높은 MFA 값을 갖게 되며 그 반대일수록 상대적으로 더욱 낮은 MFA 값을 갖게 되어 방송 효과가 낮은 데이터의 방송 중단이나 방송 빈도수를 빠르게 낮출 수 있게 된다. [보기 1]은 그 예를 보여준다.

[보기 1] : 데이터 a, b, c, d에 대한 MFA 값을 각각 10, 9, 8, 1이라고 하자. 한번에 방송될 방송 프로그램의 크기는 3 즉, 3개의 데이터만 방송한다고 하자. 또한, a는 과거에 인기가 있어 높은 MFA 값을 갖게 되었으나 최근에는 인기가 낮아지고 있는 반면에, d는 새로이 인기가 있어 계속 액세스 요청되는 데이터라고 가정하자. 이 경우에 기존 알고리즘에서는 d가 최근에 인기가 높아져 계속 액세스 요청되는 데이터라 할지라도(클라이언트의 캐시에 있지 않는 한) 여러 번의 방송 주기가 지나서 d에 대한 MFA 값이 8보다 크게 될 때 d가 방송될 수 있다. 이는 클라이언트의 대기시간을 크게 하여 성능을 저하시키게 된다. 반면에 개선한 알고리즘에 따르면, 이 경우에 a의 MFA 값은 빠르게 감소하는 반면에 d의 MFA 값은 빠르게 증가하게 되어 d가 더 이른 시기에 a를 대체하여 방송될 수 있다.

3. 캐싱 전략

데이터 방송 환경에서는 캐시 미스가 발생할 경우 전통적인 시스템 환경과는 달리 상대적으로 낮은 대역폭을 통하여 필요한 데이터를 서버로부터 제공받기 때문에 방송 알고리즘에 따라 미스에 따른 비용이 커지며 아울러 그 비용을 예측하기 어렵게 된다. 방송 프로그램의 내용을 완전히 아는 경우에는 미스 비용을 최소화할 수 있음은 당연한 사실이다. 그러나 클라이언트의 데이터 요청 변화를 반영하는 동적인 방송 환경에서는 방송 주기마다 방송 프로그램이 다를 수 있으므로 방송 프로그램을 생성하는 알고리즘의 기본 원칙을 고려하여 대체 전략을 수립하여야 미스 비용을 가능한 최소화 할 수 있게 된다. 따라서, 제안하는 효율적인 캐시 대체 전략은 히트율을 높이고 미스 비용을 최소화하기 위해 다음과 같은 직관을 기반으로 한다.

3.1 히트율을 높이기 위한 직관

(H1) : 클라이언트가 자주 액세스 요청하는 데이터 페이지는 캐시에 있어야 한다.

클라이언트가 액세스 요청을 하는 데이터는 캐시에 있거나(cache hit) 방송중이거나(air hit), 혹은 서버에게 방송 요청을 하는(pull request) 세 가지 중 한가지이다. air 히트인 경우 원하는 데이터가 방송되기를 기다려야 하며 방송 요청한 경우도 최소한 다음 방송까지 기다려야 한다. 따라서, 자주 액세스 요청되는 데이터는 캐시에 있어 히트율을 높임으로써 가장 빠른 응답시간을 얻을 수 있다.

(H2) : 클라이언트가 자주 방송 요청하는 데이터 페이지는 캐시에 있어야 한다.

캐시 미스가 일어나는 경우 클라이언트는 서버에게 데이터 방송 요청을 한다. 미스가 된 데이터는 향후에도 많이 액세스 될 가능성이 있다는 직관에 따라 액세스 요청 빈도와 별도로 방송 요청 빈도를 고려함으로써 더욱 빠르게 캐시에 존재하게 할 필요가 있다. 따라서 이러한 데이터는 빨리 캐시에 있게 함으로써 히트율을 높일 수 있게 된다.

3.2 미스비용을 최소화하기 위한 직관

(M1) : 대기 시간이 클수록 캐시에 있어야 한다.

[15]에서 미스 비용 최소화를 위해 고려한 캐시 존속 기간과 방송 횟수는 결국 대기 시간으로 귀착되므로 중복된 고려 요소가 된다. 방송 스케줄링 알고리즘에 따르면 캐시 존속 기간이 길수록 서버가 이 데이터를 방송할 가능성이 줄거나 자주 방송하지 않을 가능성이 증가하게 된다. 또한, 방송 횟수가 작은 데이터는 클라이언트의 대기 시간이 당연히 길어지게 된다. 따라서 대기시간의 고려는 이들을 충분히 반영하는 요소라 할 수 있으며 대기 시간이 큰 데이터일수록 캐시에 존재하여야 미스 비용을 줄일 수 있게 된다. 액세스 요청한 데이터가 이미 캐시에 존재하면 방송 요청을 하지 않으므로 (H2)에서 고려한 방송 요청 빈도의 낮음이 반드시 대기시간을 크게 하지는 않는다고 할 수 있다. 오히려, 방송 요청 빈도가 낮은 데이터는 빠르게 캐시에 있게 하자는 직관이므로 비트 벡터를 사용하는 방송 알고리즘의 특성을 고려할 때 상대적으로 MFA 값이 커지게 되므로 방송 요청 빈도가 낮은 데이터는 캐시에 존재하여 방송 요청을 하지 않는 데이터 보다 방송 될 가능성이 더 높다고 할 수 있다. 다른 한편으로, 방송 요청 빈도가 낮은 데이터는 방송 요청 빈도가 높은 데이터보다 방송 될 가능성이 낮아질 수 있게 되므로 대기 시간이 길어 질수 있다. 따라서, 대기 시간을 함께 고려하여 캐시에 존재할 가능성을 높여 응답 시간을 줄이기 위한 직관이 필요하게 된다.

(H1)을 반영하기 위해 각 데이터마다 클라이언트가 액세스를 요청한 횟수(R)를 사용한다. 클라이언트가 데이터 액세스를 요청할 때마다 해당 데이터의 R은 1씩 증가한다. 데이터 액세스 요청 횟수는 클라이언트의 데이터에 대한 관심 정도에 비례한다. 예를 들어, NOD나 VOD같은 모바일 서비스에서는 일정 시점에 특정 데이터들에 대한 액세스 요청이 급격히 증가하기 시작하고 난 후 점차로 감소하는 양상을 쉽게 보일 수 있다. 따라서 액세스 요청이 증가 추세에 있는 데이터와 감소 추세에 있는 데이터를 구분할 필요가 있다. 이를 위해, 데이터 액세스 요청 시점을 반영하는 최근성을 함께 고려한다. 액세스 요청 시점의 최근성을 반영하기 위해 서버가 방송한 데이터와 함께 방송되는 방송 버전 번호를 사용한다. 방송 버전 번호는 매 방송주기마다 1씩 증가된다. 따라서 액세스 요청 빈도수(Fa)는 식 (1)과 같이 유지된다. Ri는 i번째 방송주기 동안 액세스 요청된 횟수이며 Vi는 I번째 방송주기의 버전번호이다. 식 (1)로부터 액세스 요청 횟수가 같은 데이터라도 최근에 액세스 요청된 데이터일수록 버전번호가 크게 되어 큰 값을 갖게 됨을 알 수 있다.

$$Fa = \sum(Ri \times Vi) \quad (1)$$

(H2)를 반영하기 위해 각 데이터마다 클라이언트가 방송을 요청한 횟수 (B)를 사용한다. 클라이언트가 데이터 방송 요청을 할 때마다 해당 데이터의 B는 1씩 증가한다. Fa와 마찬가지로 방송 요청 시점의 최근성을 반영하기 위해 서버가 방송한 데이터와 함께 방송 되는 방송 버전 번호를 사용한다. 따라서 방송 요청 빈도수(Fb)는 식 (1)과 같이 유지된다. Bi는 i번째 방송주기 동안 방송 요청된 횟수이며 Vi는 i번째 방송주기의 버전번호이다. 식 (2)로부터 방송 요청 횟수가 같은 데이터라

도 최근에 방송 요청된 데이터일수록 버전번호가 크게 되어 큰 값을 갖게 됨을 알 수 있다.

$$Fb = \Sigma(Bi \times Vi) \quad (2)$$

(M1)을 반영하기 위하여 클라이언트가 데이터 방송 요청을 한 후 해당 데이터가 방송될 때까지 기다리는 대기시간(WT)를 사용한다. 대기 시간은 서버의 방송 상황과 직결되므로 가장 최근의 서버 상황 반영에 비중을 두는 것이 바람직하다. 따라서, 최근의 대기시간에 비중을 두어 식 (3)과 같이 평균값을 이용하여 WT를 유지한다.

$$WT = Avg(WTi) \quad (3)$$

식 (3)에서 Avg(WTi)는 i번째 방송주기까지의 평균대기 시간을 의미하며 식 (4)와 같이 순환적으로 정의된다. 식 (4)에서 WTi는 i번째 방송주기에서의 대기시간을 의미한다.

$$Avg(WTi) = (Avg(WTi-1) + WTi)/2 \quad (4)$$

이상으로부터 히트율을 높이기 위한 직관과 미스 비용을 줄이기 위한 직관을 함께 반영하여다음과 같이 캐싱 전략을 도출할 수 있다.

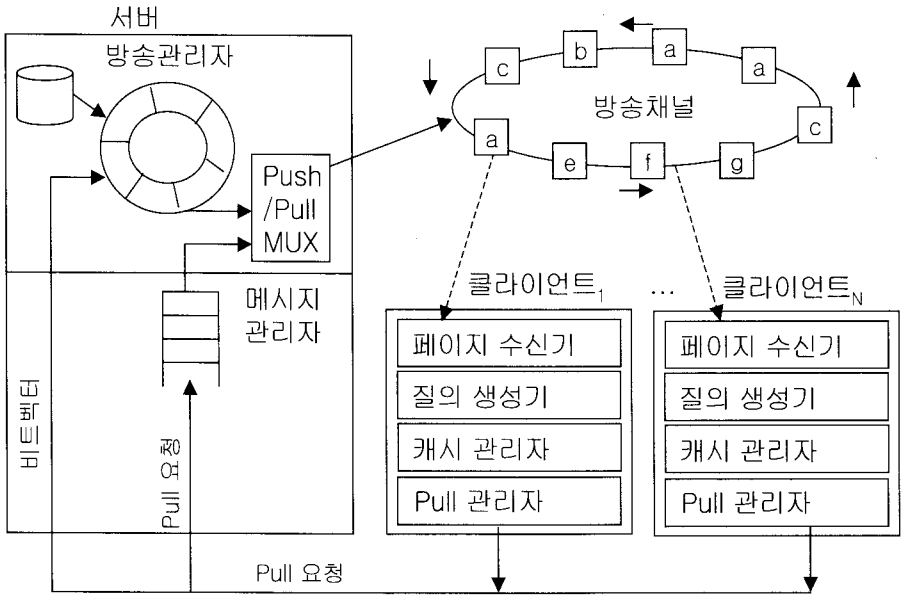
$$2FWT = Fa \times Fb \times WT \quad (5)$$

데이터 페이지 대체를 위한 희생자는 식 (5)로부터 도출된 값이 제일 작은 데이터 페이지가 된다. 식 (5)에서 최근에 액세스 요청 횟수가 많고 방송 요청 횟수가 많을수록, 그리고 대기시간이 길수록 희생자로 선택될 가능성이 줄어들게 됨을 쉽게 알 수 있다.

4. 성능 평가

4.1 성능 평가 환경

새로운 캐싱 전략을 사용하는 데이터 방송 시스템(이후 NABV : New Adaptive Bit Vector)에서 캐싱 전략의 변화가 시스템의 성능에 미치는 영향을 알아보기 위하여 전통적인 시스템에서 보편적



[그림 2] 시뮬레이션 모델

으로 사용되는 캐싱 전략인 LRU[6]를 사용하면서 비트 벡터를 기반으로 하는 데이터 방송 시스템[7](이후 BV : Bit Vector)과 비트 벡터를 기반으로 하면서 방송 스케줄링 전략과 캐싱 전략을 각각 개선한 데이터 방송 시스템[16](이후 ABV : Adaptive Bit Vector)과 성능을 평가한다. 아울러, push 기반의 방송 디스크 기법[4](이후 BC : Broad-Cast Disk)을 포함하여 평가한다. 시뮬레이션 모델은 [7]에서 제시된 것을 사용하여([그림 2] 참조) CSIM[13]을 이용하여 구현하였다. 다만, [7]의 시뮬레이션 모델에서 클라이언트가 자신이 필요로 하는 페이지를 얻을 때까지 방송채널을 감시하기 위해서 시스템 전체에 하나의 페이지 수신기 모듈을 이용하고 있지만, 본 논문에서는 각 클라이언트마다 하나의 페이지 수신기 모듈을 갖도록 수정함으로써 [7]의 연구에 비해 실세계의 작업부하를 더 현실적으로 구현하고자 노력하였다.

각 클라이언트는 페이지 수신기, 질의 생성기, 캐시 관리자, Pull 관리자의 4개의 프로세스로 구성된다. 먼저 질의 생성기에 의해 하나의 읽기 연산이 제기되면 캐시 관리자를 통해 해당 데이터가 캐시에 존재하는지 확인하게 된다. 만일 캐시에서 해당 데이터를 찾을 수 없을 경우에는 현재 방송 프로그램에 해당 데이터가 포함되어 있는지 확인하게 되고 여기에도 포함되어 있지 않을 경우에 서버에 pull 요청을 제기하게 된다. 이때 해당 클라이언트에 있는 비트 벡터도 함께 보내짐을 주목하라. 이렇게 생성된 pull 요청과 비트 벡터는 본 논문에서 제안된 기법에 따라 적절히 스케줄링 되어 방송되게 되고 자신이 필요로 하는 데이터를 획득한 클라이언트는 또 새로운 데이터 읽기 연산을 제기할 준비를 하게 된다. 각 클라이언트는 하나의 페이지를 방송이나 방송 요청을 통해 받을 때까지 새로운 페이지에 대한 연산을 제기하지 않기 때문에 시뮬레이션 중 작업의 부하는 클라이언트의 수로 일정하게 유지될 수 있으므로 폐쇄된 큐잉 모형(closed queueing model)을 채택한 것으로 볼 수 있다. <표 1>은 시뮬레이션 모델과 관련된 시스템 및 응용 매개변수를 보여준다.

데이터베이스의 데이터 항목의 수를 실제 이동 응용에서 이용되는 크기에 비해 적은 3000으로 설정하였는데 이는 상대적으로 적은 수의 클라이언

〈표 1〉 시스템 및 응용 매개변수

매개 변수	설 명	설 정 값
db_size	데이터 페이지의 수	3000으로 고정
num_clients	클라이언트의 수	90으로 고정
cache_size	클라이언트 캐시의 크기	db_size의 0.5%
queue_size	클라이언트 요청들의 대기 큐의 크기	num_clients의 10%
think_time	클라이언트의 각 요청 간 대기시간	10초
down_time	서버가 한 페이지를 방송하면 클라이언트에 도달될 때까지 소요되는 시간	0.2초
upload_time	클라이언트가 페이지를 요청하는데 소요되는 시간	0.01초
cache_acc_time	캐시에 있는 한 페이지를 액세스하는데 소요되는 시간	0.001초
bc_rate	방송 및 요청 중 방송에 할당된 채널의 비율	10%~90%, 20%씩 증가
num_bcdisk	방송 디스크의 수. 전체 데이터 페이지를 num_bcdisk 개의 그룹으로 나눔	3으로 고정
bcdisk_ratei	i번째 방송 디스크의 방송비율. 이 비율의 합이 100이어야 함	0~100사이의 값
bcdisk_acc_ratei	클라이언트가 i번째 방송 디스크에 있는 페이지를 액세스할 비율. 이 비율의 합이 100이어야 함	0~100사이의 값

트 간에 적정수준의 방송 히트율을 확보하기 위한 것이다. 반면에 캐시의 영향을 조금 더 세밀하게 관찰하기 위하여 클라이언트 캐시의 크기는 전체 데이터 항목의 0.5%로 비교적 큰 값을 선택하였다. 클라이언트는 자신이 필요로 하는 데이터가 차기 방송프로그램에 포함되어 있지 않을 경우 서버에 해당 데이터를 직접 방송 요청하여 얻게 되는데 이러한 요청이 너무 많을 경우 더 이상 방송 요청을 받을 수 없도록 하기 위해서 서버의 요청 대기 큐의 크기를 클라이언트 수의 10%로 제한하였다. 클라이언트가 제기한 읽기연산에 대해 방송이나 방송 요청에 의해 해당 데이터를 얻은 후에 일정한 시간이 경과하는 동안 새로운 연산을 제기하지 않도록 하기 위해서 think_time을 10초로 설정하였다. 서버가 방송 요청된 데이터를 클라이언트로 전송하면 각 클라이언트는 적어도 down_time 만큼의 시간, 즉 0.2초가 지나야 해당 데이터를 읽을 수 있으나 상대적으로 크기가 작은 데이터 방송요청을 서버로 전송하기 위해서는 upload_time 만큼의 시간, 즉 0.01초가 필요하다. 클라이언트와 서버간의 업로드 및 다운로드에 필요한 시간에 비하면 작은 값이지만 캐시로부터 데이터를 읽는데 소요되는 시간도 캐시 히트율의 차이를 정확히 모델링하기 위해 필요하다. 왜냐하면, 캐시 히트율이 상대적으로 높은 기법은 캐시 액세스 수가 늘어나는 반면 서버의 방송 혹은 방송요청 횟수가 줄어들지만 그 반대의 경우도 있을 수 있기 때문이다. 이러한 비용을 적절히 반영하기 위해서 [8]에서와는 달리 캐시로부터 하나의 데이터 페이지를 액세스하는 시간, 즉 cache_acc_time을 0.001초로 설정하였다.

한편, 다운로드 채널을 방송에 할당하는 비율에 따라 성능에 커다란 영향을 줄 수 있으므로 방송비율, 즉 bc_rate를 10%에서부터 20%씩 증가시키면서 실험하였다. 그리고 모든 데이터를 num_bcdisk개의 논리적인 방송디스크로 나누고 각 디스크에 있는 데이터의 방송빈도를 지정하기 위해서 bcdisk_rate1, bcdisk_rate2, bcdisk_rate3를 도입하였으며 당연히 3개의 합은 100이된다. 또한, 클라이언트 요청의 집중성(skewness)을 구현하기 위해 각 방송 디스크별 액세스 비율인 bcdisk_acc_rate1, bcdisk_acc_rate2, bcdisk_acc_rate3를 도입하였으며 역시 3개의 합은 100이 된다. 예를 들어, bcdisk_acc_rate1이 70이라면 각 클라이언트의 100개의 읽기 연산 중 70개는 방송디스크 1에 집중됨을 의미한다.

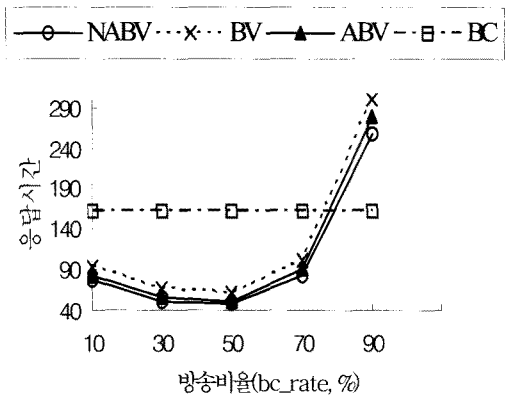
4.2 성능 평가 결과

시뮬레이션 결과의 정확성을 제고시키기 위해서 각 시뮬레이션의 초기단계에 수집된 자료는 결과에 반영하지 않았으며, 또한 각 시뮬레이션에 대해 난수생성기의 시드 값을 변경시키면서 수차례 반복 실험한 결과에 대한 평균을 이용하여 최종결과를 산출하였다. 성능 척도로는 평균 응답 시간을 사용하였으며 데이터 방송 환경에서 대기 시간의 고려가 필요함을 입증하기 위해 즉, 캐시 히트율은 적절한 성능 척도가 아님을 입증하기 위해 캐시 히트율에 대한 결과도 도출하였다. 한편, 기본적으로 혼합 기반의 데이터 방송 시스템에서는 push와 pull의 대역폭 할당과 방송 디스크간의 액세스율 등이 전체적으로 성능에 커다란 영향을 줄 수 있는 주요 매개변수들이므로 이들 변화에 대한 성능 변화를 비교한다.

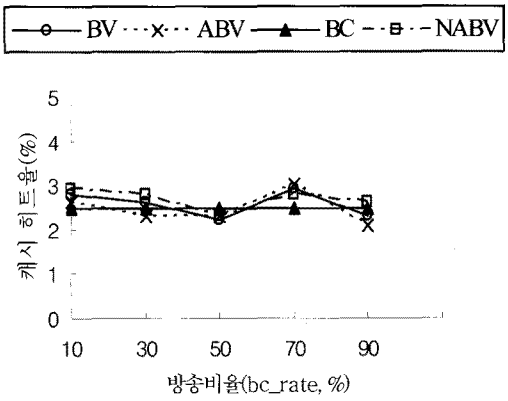
4.2.1 Push와 Pull 비율의 변화에 따른 성능

혼합 기반의 데이터 방송 시스템은 정해진 방송 알고리즘에 따라 데이터를 방송하면서 동시에 대역폭의 일정 부분을 할당받아 클라이언트로부터 직접 방송요청된 데이터를 방송하게 된다. 따라서 방송채널 할당 비율에 따른 성능의 변화를 보이기 위한 실험을 수행하였다.

[그림 3]은 방송비율(bc_rate) 변화에 따른 응답시간의 변화를 보여주고 있다. 여기에서 방송 디스크의 방송 비율은 bcdisk_rate1 = 60%, bcdisk_rate2 = 30%, bcdisk_rate3 = 10%로 고정하였으며 클라이언트가 방송 디스크의 페이지를 액세스 할



[그림 3] 방송비율 변화에 따른 응답시간

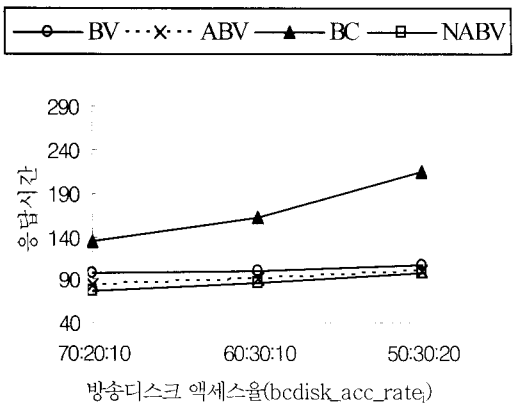


[그림 4] 방송비율 변화에 따른 캐시 히트율

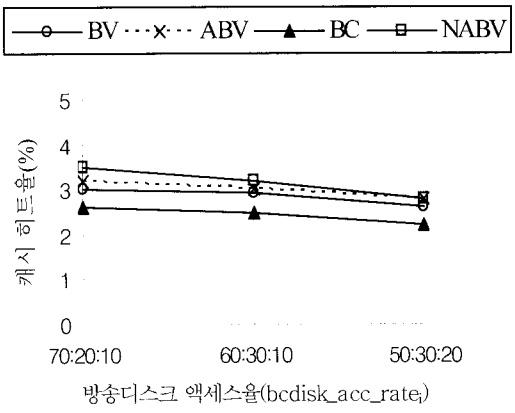
비율도 bcdisk_acc_rate1 = 60%, bcdisk_acc_rate2 = 30%, bcdisk_acc_rate3 = 10%로 고정하였다. BC는 클라이언트의 요청에 따른 방송을 하지 않으므로 일정한 응답시간을 보이고 있으나, BV와 ABV는 방송 비율에 따라 응답시간의 변화를 보이고 있다. 전반적으로 ABV가 BV보다 방송 비율에 상관없이 좋은 성능을 보이고 있음을 볼 수 있다. 이는 ABV가 클라이언트의 액세스 요청 변화를 반영한 캐시 대체 전략을 사용한 결과로 판단된다. 그리고 BV와 ABV가 방송 비율이 낮을수록 BC보다 좋은 성능을 보이고 있으나 방송 비율이 증가할수록 성능 차이가 줄어들다가 방송 비율 80 부근부터는 오히려 성능이 떨어지고 있음을 알 수 있다. 이는 클라이언트의 데이터 액세스 요청 패턴 변화에 능동적으로 대처할 수 있는 대역폭의 할당이 절대적으로 부족해짐에 따라 극심한 대역폭 경쟁으로 인해 pull을 위해 할당된 대역폭의 효용성은 없으면서 결과적으로 push를 위한 대역폭의 감소만 초래하여 필요한 데이터를 다음 방송 시까지 기다려야 하는 경우가 발생되어 전체적으로 응답시간의 증가를 초래하는 것으로 판단할 수 있다. 반대로 40% 부근까지 방송 비율이 낮아질수록 응답시간이 감소하는 것은 pull을 위해 할당된 대역폭이 클라이언트의 데이터 액세스 요청 패턴 변화에 능동적으로 대처할 수 있게 됨으로써 할당의 효용성이 반영된 결과로 분석된다. 그러나 그 이하 에서는 별다른 성능 향상을 보이지 못하는 것은 주어진 실험 환경에서는 더 이상의 pull을 위한 대역폭 할당은 효과가 없음을 의미한다고 할 수 있다.

한편, NABV는 ABV와 비슷한 성능을 보여주고 있어 미스 비용을 줄이기 위해 ABV에서 처럼 여러 가지 요소를 중복하여 고려하지 않고 이들과 직접적으로 연관 있는 대기시간만을 고려하여도 성능에는 차이가 없음을 알 수 있다. 이런 결과는 또한 클라이언트의 액세스 요청 빈도와 방송 요청 빈도를 구분함으로써 방송을 요청 하여 방송된 데이터가 좀 더 빨리 캐시에 존재할 수 있게 되어 극심한 대역폭 경쟁이 다소 완화될 수 있기 때문으로 분석할 수 있다.

[그림 4]는 방송비율 변화에 따른 캐시 히트율의 변화를 보여주고 있다. 캐시 히트율의 변화에 대해 일관성 있는 분석이 어려움을 알 수 있다. 즉, 히트율이 낮아도 응답시간은 적게 나타날 수 있으며 반대로 히트율이 높아도 응답시간은 크게 나타날 수 있음을 볼 수 있다. 이는 데이터 방송 시스템에서는 히트율뿐만 아니라 미스에 따른 대기시간이 성능에 영향을 미치고 있음을 입증해고 있다. 예를 들어, 히트율이 높아도(낮아도) 대기시간이 길어(짧아)짐으로서 전체적으로 응답시간이 길어(짧아)질 수 있음을 확인할 수 있다. 따라서 전통적인 시스템에서처럼 캐시의 히트율이 성능평가의



[그림 5] 방송디스크 액세스율 변화에 따른 응답시간



[그림 6] 방송디스크 액세스율 변화에 따른 캐시 히트율

지표로 적절하지 않음을 입증하고 있다.

4.2.2 방송 디스크 액세스율의 변화에 따른 성능

방송 디스크 기법을 도입하는 가장 큰 목적은 클라이언트의 데이터 액세스 빈도에 따라 방송 디스크간의 방송 횟수를 차별화함으로써 성능 향상을 얻고자 하는 것이다. 따라서 방송 디스크 사이의 액세스율이 방송 디스크 사이의 방송 횟수와 유사할수록 좋은 성능을 보이게 된다.

[그림 5]는 방송 디스크 사이의 액세스율에 따른 응답시간의 변화를 보여준다. 여기에서 방송 디스크 비율은 bcdisk_rate1 = 60%, bcdisk_rate2 = 30%, bcdisk_rate3 = 10%로 고정하였고 bc_rate = 70%로 하였다. 클라이언트의 데이터 요청이 디스크 사이에 고르게 분산될수록 전체적으로 응답시간이 증가하고 있다. 이는 방송 디스크 사이의 액세스율 차별화를 통하여 성능을 향상하고자 하는 방송 디스크 기법의 기본 목적과 배치되는 경우이므로 당연한 결과로 보여 진다. 특히, BC는 분산될수록 응답시간의 변화폭이 커지는 반면에 BV와 ABV는 그렇지 않음을 볼 수 있으며 이는 BC의 경우 미스가 발생하는 경우 다음 방송 시까지 기다려야하므로 pull을 위한 대역폭을 사용할 수 있는 BV나 ABV보다 대기 시간이 길어진 결과로 해석할 수 있다. 한편, 전반적으로 클라이언트의 요청을 능동적으로 반영할 수 있는 BV와 ABV가 BC보다 좋은 성능을 보이고 있으며 스케줄링 전략과 캐싱 전략을 개선한 AVB가 BV보다 다소 좋은 성능을 보여주고 있어 전략의 개선이 더 능동적으로 클라이언트의 요청 변화에 대처하는 데 기여함을 알 수 있다. 한편, 클라이언트의 요청 패턴 변화를 복합적으로 고려하면서도 캐싱 전략의 고려요소를 단순화시킨 NAVB도 ABV와 비슷한 성능을 보이고 있어 미스비용의 고려에 있어 대기시간만을 고려해도 충분함을 알 수 있다. 아울러, 캐시 히트율도 [그림 6]에서 볼 수 있는 바와 같이 각 방송 디스크 방송(60 : 30 : 10)과 차이를 보여 데이터 요청이 각 디스크로 분산될수록 전체적으로 떨어지고 있다

5. 결 론

이동 컴퓨팅 환경에서 다수의 클라이언트에게 효과적으로 이동 데이터 서비스를 제공하기 위한 방법의 하나로 데이터 방송이 주목을 받고 있다. 데이터 방송이 효율적이려면 클라이언트의 데이터 액세스 요청 사항 변화를 반영하는 것이 바람직하다. 이를 위해, 본 논문에서는 방송 알고리즘의 특성을 반영하여 클라이언트의 데이터 방송 요청 빈도수를 캐싱 전략에 고려하였고 미스 비용을 줄이기 위해 대기시간만을 고려하는 것으로 단순화 하

였다. 개선된 방송 시스템의 성능 평가를 위해 기존에 제안된 시스템과의 평가 결과에 따르면, 클라이언트의 액세스 요청 패턴이 동적으로 변하는 환경에서는 당연히 클라이언트의 요청 변화를 능동적으로 수용할 수 있는 혼합 기반의 시스템이 우수한 성능을 나타내고 있다. 전반적으로 클라이언트의 액세스 요청 패턴을 반영하여 제안한 전략이 응답시간 측면에서 다소 좋은 성능을 보이고 있다. 그러나 성능의 차이가 크지 않게 나타난 것은 캐싱 전략에서 미스 비용을 최소화 하기 위해 중복 특성이 있는 여러 가지 요소를 고려하는 대신 대기 시간만을 고려해도 충분함을 입증해 주고 있다. 아울러, 혼합 데이터 방송 시스템에서는 push를 위한 데이터 방송 알고리즘이 시스템의 성능에 캐싱 전략보다 더 큰 영향을 미치고 있음을 말해 주고 있다.

참 고 문 헌

[1] D. Aksoy and M. Franklin, "RxW : A Scheduling Approach for Large-Scale On-Demand Data Broadcast", 「IEEE/ACM Transactions on Networking」 Vol.7, No.6(1999), pp.846-860.

[2] S. Acharya, M. Franklin, and S. Zdonik, "Balancing Push and Pull for Data Broadcast", 「Proc. of ACM SIGMOD」, (1997), pp.183-194.

[3] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast Disks : Data Management for Asymmetric Communication Environments", 「Proc. of ACM SIGMOD」, (1995), pp.199-210.

[4] D. Barbara, "Mobile Computing and Databases-A Survey", 「IEEE Transactions on Knowledge Engineering」, Vol.11, No.1(1999), pp.108-117.

[5] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments", 「Proc. 6th Int. Conf. Mobile Computing and Networking」, (2000), pp.200-209.

[6] S. Galvin and P. B. Galvin, "Operation System Concepts", 4th Edition, 「Addison Wesley」, 1994.

[7] Q. Hu, D. L. Lee, and W. C. Lee, "Dynamic Data Delivery in Wireless Communication Environment", 「Workshop on Mobile Data Access」, (1998), pp.213-224.

[8] A. Kahol, S. Khurana, S. K. S. Gupta, and P. K. Srimani, "A Strategy to Manage Cache Consistency in a Disconnected Distributed Environment", 「IEEE Trans. on Parallel and Distributed Systems」, Vol.12, No.7(2001), pp.686-700.

[9] V. Kumar, *Mobile Database System*, Wiley-InterscienceK, 2006[13] S. K. Madria and B. K. Bhargava, "A Transaction Model to Improve Data Availability in Mobile Computing", 「Distributed and Parallel Databases, Vol.10, No.2(2001), pp.127-160.

[10] Y. Lai, Z. Tari, and P. Bertok, "Cost Efficient Broadcast based Cache Invalidation for Mobile Environments", 「Proc. of ACM Symposium on Applied Computing」, (2003), pp.871-877.

[11] S. Lee, C. Hwang, and M. Kitsuregawa (2006), "Efficient, Energy Conserving Transaction Processing in Wireless Data Broadcast", 「IEEE Trans. On Knowledge and Data Engineering」, Vol.18, No.9(2006), pp.1225-1238.

[12] J. B. Lim and A. R. Hurson, "Transaction Processing in Mobile, Heterogeneous Database Systems", 「IEEE Trans. On Knowledge and Data Engineering」, Vol.14, No.6(2002), pp.1330-1346.

[13] H. Schwetman, "CSIM User's Guide for Use with CSIM, Revision 16", 「Microelectronics and Computer Technology Corporation」, 1992.

[14] X. Shao and Y. Lu, "Maintaining cache Consistency of Mobile Database Using Dynamiccal Periodical Broadcast strategy",

「Int. Conf. on Machine Learning and Cybernetics」, (2003), pp.2389-2393.

[15] D. C. Shin, "A Caching Strategy Considering Characteristics of Broadcast algorithm in Hybrid-based Data Broadcast Systems", 「Journal of Korea Information Processing Society」, Vol.12-C, No.2(2005), pp.243-250.

[16] P. Triantafillou, R. Harpantidou, and M. Paterakis, "High Performance Data Broadcasting : A Comprehensive Systems' Perspective", 「LNCS 1987, Springer-Verlag」, (2001), pp.79-90.

[17] J. Xu, Q. Hu, W.-C. Lee, and D. L. Lee, "Performance Evaluation of an Optimal Cache Replacement Policy for Wireless Data Dissemination", 「IEEE Transactions on Knowledge and Data Engineering」, Vol.16, No.1(2004), pp.125-139.

◈ 저 자 소 개 ◈

신 동 천 (dcshin@cau.ac.kr)

서울대학교 컴퓨터공학과를 졸업한 후 한국과학기술원 전산학과에서 석사와 박사학위를 각각 취득하였다. 그 후, 한국전산원 선임연구원을 거쳐 중앙대학교 정보시스템학과에서 교수로 재직중이다. 최근의 주요 관심분야는 이동 데이터베이스와 이동 서비스 모델링이다.