# 무선 클라이언트-서버 환경에서 이동 클라이언트의 캐시 데이타 재사용율 향상기법

## (Increasing a Mobile Client's Cache Reusability in Wireless Client-Server Environments)

이 송 이 †

(Song-Yi Yi)

**요 약** 무선 클라이언트-서버 환경에서 데이타 브로드캐스팅은 효율적인 데이타 보급방식이다. 서버가 데이타를 방송하면 그 중 일부를 클라이언트가 캐시에 보관하여, 낮은 통신 대역폭과 한정된 자원을 극복하고 데이타 접근 시간 등을 절약할 수 있다. 서버는 또한 무효화보고서를 방송하여 서버 데이타와 클라이언트가 캐싱한 데이타간의 일관성을 유지할 수 있도록 한다. 기존의 캐시 일관성 문제를 해결하기 위한 대부분의 연구는, 접속 단절 시간이 너무 길어서 수신하지 못한 무효화 보고서의 수가 일정 한도(윈도우 크기)를 넘으면 단순히 캐시 전체를 삭제하는 방법을 사용한다. 이 논문은 이러한 긴 접속 단절 시간의 경우에도 이동 클라이언트의 캐시 재사용율을 높일 수 있는 기법을 제시한다. 캐시의 일관성 여부에 관계없이 무조건 전체 캐시 내용을 지우는 대신, 클라이언트는 모든 데이타를 지우는 비용과 선택적으로 지우는 비용을 추측한다. 만일 모든 데이타를 지우는 비용이 높아지는 경우에는 클라이언트는 캐시를 유지하고 검증 요청을 위해 업링크를 사용하여 유효하지 않은 데이타만을 선택적으로 지운다. 이 방법은 캐시 유지 비용을 추측할 때 캐시된 데이타의 갱신 비율과 방송 빈도들을 함께 고려하므로, 효율적으로 캐시 재사용율을 높일 수 있다는 것을 모의 실험 결과를 통해 증명한다.

**키워드** : 무선 클라이언트-서버 환경, 데이타 브로드캐스트, 캐시 일관성, 무효화 보고서, 캐시 재사용율

***Abstract*** In a wireless client server environment, data broadcasting is an efficient data dissemination method; a server broadcasts data, and some of broadcasted data are cached in a mobile client's cache to save the narrow communication bandwidth, limited resources, and data access time. A server also broadcasts invalidation reports to maintain the consistency between server data and a client's cached data. Most of existing works on the cache consistency problems simply purge the entire cache when the disconnection time is long enough to miss the certain amount(window size) of IRs. This paper presents a cache invalidation method to increase mobile clients' cache reusability in case of a long disconnection. Instead of simply dropping the entire cache regardless of its consistency, a client estimates the cost of purging all the data with the cost of selective purge. If the cost of dropping entire cache is higher, a client maintains the cache and selectively purge inconsistent data using uplink bandwidth for validation request. The simulation results show that this scheme increases the cache reusability since it effectively considers the update rates and the broadcast frequencies of cached data in estimating the cost of cache maintenance.

**Key words** : wireless client-server environment, data broadcast, cache consistency, invalidation report, cache reusability

## 1. 서론 및 연구의 필요성

무선통신 기술의 발전과 휴대용 컴퓨터의 보급은 네트워크 클라이언트의 자유로운 이동성을 지원하게 되었다. 이동 클라이언트가 서버와 대화하기 위해 사용하는 무선통신망은 클라이언트에게 이동의 자유를 보장하지만, 유선보다 상대적으로 불안정하고, 대역폭이 낮고, 통

· 이 논문은 2004년도 한국학술진흥재단의 지원에 의하여 연구되었음 KRF-2004-003-D00370

† 정 회 원 : 서울대학교 BK21정보기술사업단 교수
yis@snu.ac.kr

논문접수 : 2005년 10월 5일
심사완료 : 2006년 1월 19일

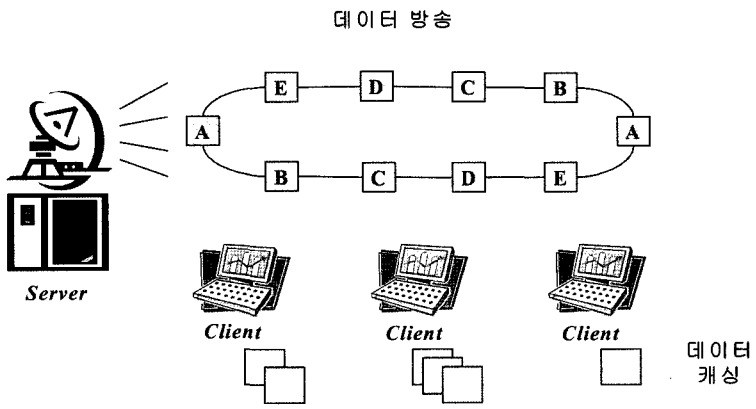


그림 1 데이타 방송과 이동 클라이언트의 데이타 캐싱

신비용이 높으므로 이를 극복할 새로운 분산시스템 기술을 필요로 한다. 캐싱은 무선 환경에서 이동 컴퓨터의 전력소모와 통신 대역폭을 감소시키기 위해 매우 중요한 기술이다. 서버가 클라이언트에게 낮은 통신비용으로 데이타를 전송하기 위해, 그림 1과 같이 자주 요청되는 데이타를 무선 상에 방송(broadcast)하는데, 이 때 클라이언트는 방송 채널을 듣고 있다가 필요한 데이타를 캐싱 함으로써 대역폭을 절약하고 고비용의 업링크(up-link) 사용을 자제 할 수 있다.

고정된 클라이언트의 캐시는 안정적이고 빠른 통신선을 이용해 서버와 항상 연결되어 있으므로, 기존 분산시스템에서 캐시의 무결성을 보장하기 위한 캐싱 기법을 고안할 때 통신선의 대역폭이나 에너지 소모는 고려하지 않았다. 그러나 이동 클라이언트는 이동 컴퓨터의 특성상 서버와의 연결 해제를 자주 반복한다. 그림 2는 연결 해제로 인해 갱신된 데이타 값을 수신하지 못해 캐싱된 데이타 값이 더 이상 유효하지 않은 예를 보여준다. 연결 해제는 이동 클라이언트가 고의적으로 이동을 하기위해 전력을 끄거나, 전력을 절약하기 위해 연결을 해제할 수도 있지만 이동 클라이언트가 사용하는 무선망이 불안정하여 이동이나 전파의 간섭 등으로 인해 일시적으로 연결이 해제 될 수도 있다. 따라서 클라이언트와 서버의 연결 해제 사유와 기간이 매우 다양하므로, 모든 경우에 대하여 캐시의 일관성 검사를 최적화할 수 있는 방법을 고안하기는 매우 어렵다.

무선 환경에서 이동컴퓨터의 캐시일관성을 효율적으로 유지관리하기 위한 여러 가지 기법들이 고안된 바 있다. 그러나 이들 기법은 대부분 브로드캐스트 타임스탬프 기법[1]을 기초로 하여 이동 클라이언트가 네트워크와 연결이 해제된 기간의 길이만을 고려하여 캐시의 일관성을 판단한다. 그러나 만일 자주 갱신되지 않는 데이타가 이동 클라이언트에 캐싱된 경우, 오랜 시간 연결

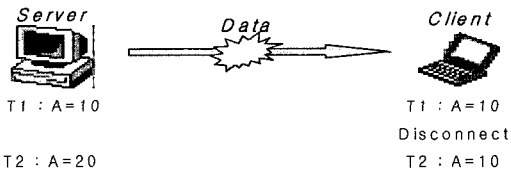


그림 2 캐시 데이타가 더 이상 유효하지 않은 경우

이 해제된 후 다시 네트워크에 접속했을 때 캐싱된 데이타의 값이 유효할 수도 있다. 기존의 기법들은 단지 네트워크 연결이 해제되었던 기간이 길다는 이유만으로 (정확히 말하면, 연결 해제되었던 시간이, 설정된 무효화 보고서 수신 윈도우 크기보다 크다는 이유만으로) 클라이언트 캐시 내용의' 유효 여부를 검사하지 않고 무조건 버린다. 캐시의 내용을 모두 버리게 되면, 브로드캐스트 채널에서 데이타를 기다리는 무선 환경상의 이동 클라이언트는, 원하는 데이타에 다시 접근하기 위해 오랜 기간 브로드캐스트채널을 들어야만 할 수 있다. 만일 재접속 후 무조건 캐시에서 삭제한 데이타가 갱신율이 매우 낮아 그동안 갱신되었을 확률이 매우 낮고, 그 크기가 매우 크며, 또한 방송 빈도도 매우 낮다면, 데이타 접근 시간과 자원 소모 면에서 이동 클라이언트는 많은 손해를 보게 된다.

이 논문은 캐시 재사용율을 높이기 위해, 기존의 클라이언트 중심 캐시 유효 검사와는 달리 서버와 클라이언트가 협력하여 캐시의 일관성을 검증하는 방법을 제안한다. 클라이언트와 서버가 협력하여 캐시의 내용을 검증하면, 보다 정확하게 캐시의 유효성을 판단할 수 있다. 단, 클라이언트가 서버에게 전달하는 업링크 메시지의 사용을 최소화하여 설계해야 한다. 업링크의 사용은 이동 클라이언트의 전력소모와 대역폭 사용의 제한 때문에 다운링크보다 상대적으로 불리하기 때문이다.

## 2. 관련 연구

현재까지 클라이언트 캐시 일관성 유지 기법에 관한 여러 연구가 진행되고 있다. 이 들 연구는 여러 가지 기준으로 분류된 바 있다. 그 중 한 가지 방법은 비동기식(asynchronous)기법과 동기식(synchronous)기법이다. 비동기식 일관성 유지 기법은 서버의 데이타가 갱신되었을 때 곧바로 캐시 무효 메시지(invalidation message)를 방송하는 방법이고, 동기식 기법은 무효화 보고서(invalidation report)를 일정한 간격으로 주기적으로 방송하는 방법이다.

또 다른 분류 방법은 서버가 클라이언트의 캐시 상태에 관한 정보를 알고 있는지(stateful), 모르는지(stateless)에 따라 나누는 것이다. 스테이트풀(stateful) 방식은 서버에서 클라이언트 캐시에 관한 정보를 유지하기 위해 특정 데이타를 캐시에 저장할 때마다 서버에 그 정보를 전송해야 한다는 면에서 대역폭의 낭비를 초래하므로 이동이 빈번한 이동 클라이언트에는 부적절하다.

이 논문은 기존의 캐시 일관성 유지 방안들을 일관성 검사의 주체에 따라 표 1과 같이 분류하였다. 즉 클라이언트가 주체인지, 서버가 주체인지, 혹은 클라이언트와 서버가 협력하여 유효성 검사를 하는지에 따라 특징을 정리하고 대표적인 검사 기법을 나열하였다. 표 1에서 보듯이, 각 분류에 따라 캐시 검사를 위한 통신비용과 업링크 사용 비용, 캐시 재사용율이 다르다. 대부분의 클라이언트 중심 캐시 일관성 유지 기법은 서버가 주기적으로 서버에서 일정 시간 동안 갱신 된 데이타들에 대한 정보(무효화 보고서 Invalidation Report)를 브로드캐스트 하는 방법을 기본으로 한다. 이 기법은 대역폭의 오버헤드를 줄일 수 있어 이동 클라이언트 캐시 일관성 유지를 위해 가장 널리 사용되는 방법이다. 하지만 무효화 보고서를 수신하지 못하는 경우에는 캐시의 유효성을 알지 못하므로 무조건 캐시의 모든 데이타를 버려야 한다. 즉, 서버에서의 데이타 갱신 여부에 따라 클라이언트의 캐시 일관성 유지 여부가 결정되기 보다는 이동 클라이언트의 접속 단절된 시간에 따라 캐시 유효 여부가 결정된다. 서버에서 데이타 갱신이 발생하지 않은 경우라 하더라도 오랜 시간 접속 단절된 클라이언트는, 비록 캐싱한 데이타가 계속 유효할지라도, 단절된 시간이 허용된 무효화 보고서 윈도우 크기보다 크면 캐시의 내용을 무조건 버리게 되어있다. 기존의 대부분 무선 클라이언트 캐시는 비싼 통신비용 때문에 (특히 높은 업링크 사용 비용 때문에) 클라이언트가 서버와의 통신 없이 자신의 캐시 유효성을 접속단절 시간에 비추어 예측하였다. 따라서 캐시의 재사용율에 관한 연구는 아직까지 거의 수행된 바 없다.

이 논문은 캐시의 재사용율을 높이기 위해 클라이언트와 서버가 협력하여 유효성을 판단하는 기법을 제시한다. CCI(Cost-Based Cache Invalidation)[5]와 ECI[7]는 이 연구의 선행 연구로 발표된 것이다. CCI에서 클라이언트는 연결이 재개 된 후 캐시의 상태를 서버에게 전송한다. 서버는 캐시를 유지하는데 필요한 비용과 캐시 내용을 전부 버리는 비용을 모든 이동 클라이언트에 대해 일괄적으로 계산하여 총 비용이 낮은 결정을 클라이언트에게 전송한다. 이 방법은 연결 해제 시간외에 캐시 유지비용을 고려하였다는 장점이 있으나 여러 가지

표 1 관련 연구 분류 및 특징

| 캐시 일관성 검사 주체에 따른 분류 | 클라이언트 중심 방식 | 서버 중심 방식 | 기존 클라이언트-서버 방식 | 제안하는 클라이언트-서버 방식 |
|---|---|---|---|---|
| 기본 아이디어 | 브로드캐스트 타임스탬프와 무효화 보고서 사용 | 서버가 클라이언트 캐시의 유효성 판단 | 클라이언트와 서버가 협력하여 일관성 유지 | 클라이언트와 서버가 협력하여 일관성 유지 |
| 캐시 유효성 판단 방법 | 캐시 유효성 판단 시 연결 해제 시간만을 고려 | 클라이언트는 서버에게 캐시 데이타의 검증 문의 | 서버 중심 기법에 가까움 | 캐시 재사용율 향상을 위해 데이타의 특성(방송 빈도, 갱신 주기 등) 활용 |
| 특징 | 캐시 검증을 위한 업링크 사용 자제 | 많은 통신량과 업링크 사용으로 인한 통신 오버헤드 심각 | 제안된 기법이 별로 없음 | 업링크 메시지 사용을 최소화하기 위해 BT 기반 유지 |
| 장점 | 통신비용이 적어 이동 클라이언트 환경에 적합 | 캐시 재사용율 높음 | 통신비용은 클라이언트 중심 기법보다 높고 캐시 재사용율은 서버중심 기법보다 높지 않지만 선택적 캐시 검증 가능 | 낮은 통신비용을 유지하면서 캐시 재사용율 향상에 중점을 둠 |
| 단점 | 장기간의 접속 단절 시 캐시 재사용율 매우 낮음 | 통신 오버헤드, 서버의 과부하, 확장성(scalability) 낮음 | | |
| 관련 연구 | BT[1]<br>BGI[2]<br>BS[3]<br>DRCI[4] | CCI[5] | GCORE[6] | ECI[7] |

개선되어야 할 사항이 있었다. 우선 대역폭 오버헤드 중 클라이언트에서 서버로 정보를 전달하는 업링크 통신비용은 다운링크 통신비용보다 매우 높다는 사실을 고려하지 않았다. 또한 캐시를 유지하는 비용을 전체 클라이언트에 대한 총 비용으로 일률적으로 계산하여, 서버에서 데이타 갱신율이 높을 경우, 이동 클라이언트에게 일률적으로 캐시를 버리라는 메시지가 전달되어 클라이언트 캐시의 내용에 관계없이 캐시 재사용율이 급격하게 떨어지는 단점이 있다. 또한 서버가 모든 클라이언트의 캐시 유효/무효 판단을 하므로 확장성에 문제가 있었다. 클라이언트의 수가 매우 많을 때, 서버의 응답을 기다리는 병목현상이 발생한다.

ECI는 캐시의 유효 판단을 클라이언트가 하도록 CCI를 개선한 방법으로, 비용함수를 클라이언트가 예측한다는 측면에서 이 논문에서 제안하는 기법과 비슷하다. 하지만 ECI의 비용함수는 데이타의 특성을 충분히 반영하지 못했고, 다양한 변수에 대한 모의실험이 부족하였다. 이 논문에서는 연결해제 시간과 함께 서버의 데이타 갱신율과 방송 빈도를 고려하여 오랜 기간동안 갱신되지 않는 데이타를 캐싱한 이동 클라이언트의 캐시 재사용율을 높임으로써, 즉, 캐시에서 버려지는 유효한 데이타의 수를 최소화함으로써, 통신대역폭의 사용과 이동 컴퓨터의 전원소모를 줄이고자 한다.

## 3. 데이타 브로드캐스트 시스템

이 장에서는 먼저 브로드캐스트 시스템의 기본 구조를 보이고, 제안하는 기법이 기반으로 하고 있는 IR 기반 캐시 데이타 유효성 확인 기법을 설명한다.

### 3.1 브로드캐스트 시스템 기본 구조

서버는 데이타베이스내의 데이타들을 주기적으로 방송한다. 각 데이타들은 인기도나 중요도에 따라 방송 빈도(Freq)가 다르며, 모든 데이타가 최소한 한 번 방송되는 기간을 주 방송 사이클(Major Broadcast Cycle)이라고 한다. 즉, 데이타 $d_x$의 방송 빈도 $Freq_x$는 주 방송 사이클 안의 방송 횟수를 의미한다. 주 방송 사이클은 여러 개의 부 방송 사이클(Minor Broadcast Cycle)로 나눌 수 있다.

서버는 부 방송 사이클 후에 일정 주기(L)마다 일정 기간 동안 갱신된 데이타들의 이름(Data ID)과 갱신 시간(Update Timestamp)의 짝($d_x$, $t_x$)을 방송한다. 이것을 무효화 보고서(Invalidation Report)라고 하고, 이동 클라이언트는 서버에 질의하지 않고도 무효화 보고서를 통해 캐싱된 데이타들의 유효성 여부를 판단할 수 있다. 무효화 보고서에 포함된 데이타를 캐싱하고 있는 이동 클라이언트는 해당 데이타가 서버에서 변경되어 더 이상 유효하지 않음을 알게 되므로 캐시 데이타 중 무효화 보고서에 포함된 데이타를 유효하지 않은 것으로 간주하고 브로드캐스트 채널에서 다시 수신한다. 이동클라이언트에서, 브로드캐스트채널로부터 데이타를 캐시하지 못한 데이타나 무효화보고서에 의해 캐시에서 유효하지 않은 것으로 판명된 데이타에 대해 질의가 발생하는 경우 업링크 채널을 이용하여 데이타를 요청한다. 서버는 데이타의 요청을 큐에 받아 방송 스케줄에 반영하므로 많은 클라이언트로부터 데이타 요청이 발생한 데이타는 방송 빈도가 높아지게 된다. 그림 3은 이러한 기본 시스템 구조를 나타낸 것이다.

브로드캐스트 채널의 마이너 브로드캐스트 사이클은 데이타 이름(Data ID)과 브로드캐스트 빈도(Freq), 데

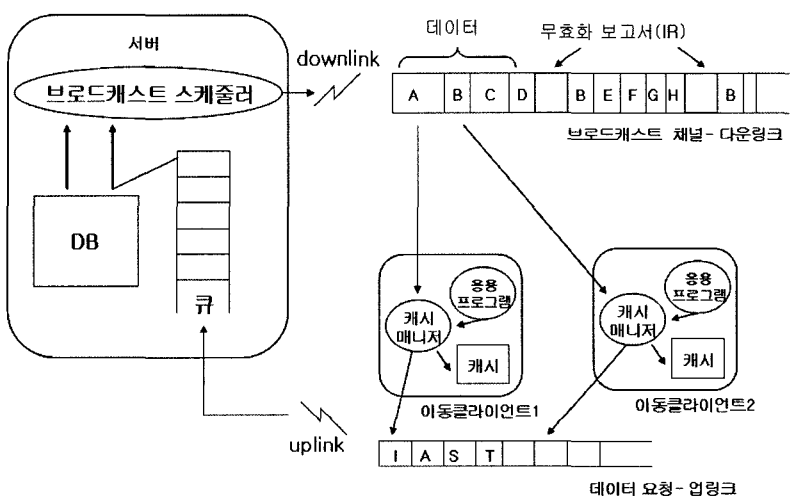


그림 3 시스템 구조

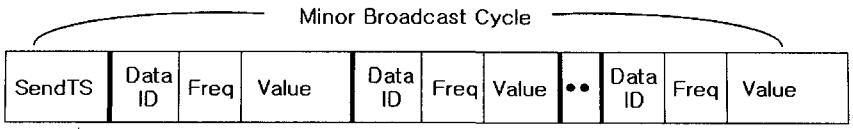


그림 4 브로드캐스트 채널(마이너 브로드캐스트 사이클)의 구조

이타 값을 함께 방송한다(그림 4). 또한 마이너 브로드캐스트 사이클 방송 시점의 타임스탬프인 *SendTS*를 방송하여 클라이언트가 데이타 값과 함께 방송 시점의 타임스탬프를 기억하도록 한다. 데이타의 값은 데이타에 따라 그 크기가 다르다고 가정한다.

이동 클라이언트의 캐시는 가장 최근에 받은 무효화 보고서의 타임스탬프 *LastTS*를 저장한다. 이 값을 브로드캐스트 채널로부터 가장 최근에 수신된 무효화보고서의 타임스탬프 값과 비교함으로써 접속 단절 기간을 판별한다. 캐시에 데이타를 저장할 때는 브로드캐스트 채널로부터 획득한 데이타의 ID, 방송 빈도, 데이타 크기와 데이타 값을 *UpdateTS* 항목과 함께 저장한다(그림 5). 여기서 데이타 값은 실제 데이타의 값을 의미할 수도 있지만 데이타가 저장된 기억 장소의 주소 값을 의미한다고 가정한다.

*UpdateTS* 항목은 해당 데이타가 갱신된 시점의 타임스탬프이다. 데이타가 브로드캐스트 채널에서 수신되어 캐시에 저장될 때는 함께 수신된 *SendTS*가 이 값이 되지만, IR에서 해당 데이타의 갱신 사실이 $(d_x, t_x)$로 수신되는 경우, 갱신 시간 $t_x$이 $UpdateTS_{dx}$에 저장된다. 따라서 $UpdateTS_{dx} > LastTS$인 캐시 데이타는 더 이상 유효하지 않은 데이타이고 $UpdateTS_{dx} \leq LastTS$를 만족해야만 유효한 데이타가 된다.

클라이언트는 또한 캐싱된 데이타가 무효화 보고서에 얼마나 자주 포함되는지를 *IRRate* 항목을 통해 기록한다. 이것은 각 데이타의 갱신율을 추정하는데 사용한다. 즉 *IRRate*란 전체 수신된 IR수 중에서 해당 데이타가 포함된 IR의 수를 의미하는데 이 데이타가 얼마나 자주 갱신되는지 추측하는데 사용할 수 있다. IR은 일정기간 동안에 갱신된 데이타들의 이름과 갱신 시간을 포함하고 있다. IR은 주기적으로 방송되므로 실제 갱신율과

| | Data ID | Freq | Data Size | Value | UpdateTS | IR Rate |
|---|---|---|---|---|---|---|
| LastTS | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

그림 5 브로드캐스트 채널(마이너 브로드캐스트 사이클)의 구조

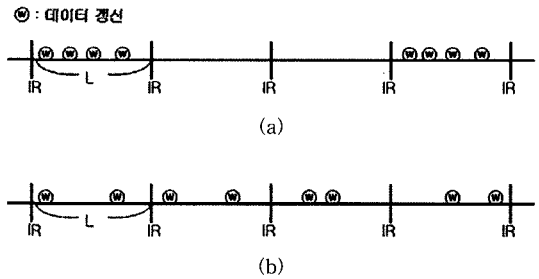


그림 6 데이타의 갱신율과 IR방송률

IR을 통해 방송되는 비율은 일치 하지 않는다. 그러나 IR에 많이 포함되는 데이타일수록 그렇지 않은 데이타보다 갱신 확률이 높다고 볼 수 있다.

이 논문에서는 데이타의 갱신 발생이 포아송 분포를 따른다고 가정한다. 그림 6의 두 그림은 모두 어떤 데이타에 대해 일정 기간동안 8번의 갱신이 일어난 시점을 표시한 것이다. 그림 6의 (a)번 그림은 4번의 IR 수신 중 두개의 IR에 해당 데이타가 포함되는 반면 (b)번은 4번의 IR 수신 중 4번의 IR에 해당 데이타가 포함된다. 첫 번째 그림은 한번의 IR 방송 주기 안에서 여러 번의 데이타 갱신이 발생하므로, 실제 갱신율보다 클라이언트가 추정하는 갱신율은 매우 낮다. 두 번째 그림도 클라이언트가 추정하는 갱신율은 실제 갱신율보다 낮지만, 클라이언트는 IR 방송주기에 맞추어 갱신 사실을 통보받으므로 IR 방송 주기(L)보다 짧은 갱신 주기는 클라이언트 입장에서 별다른 차이가 없다. 클라이언트가 추측한 갱신 주기로 L 시간 경과 후 데이타가 유효하지 않음을 짐작하기에 충분하다. 이 논문의 실험에서는, 데이타 갱신 주기에 편차가 심하지 않고, 그림 6의 (b)와 같이 거의 일정한 주기로 갱신이 일어난다고 가정하였다.

### 3.2 브로드캐스트 환경에서 IR기반 데이타 일관성 유지

IR을 사용한 캐시 일관성 유지 기법은, 이동 클라이언트가 캐시의 유효성 확인을 위해 서버와 통신할 필요 없이 서버가 주기적으로 무효화 보고서를 사용함으로써 서버의 성능이 이동 클라이언트의 수나 데이타의 갱신율 등에 영향을 받지 않는다. 또한 이동 클라이언트는 캐시 일관성 유지를 위해 업링크를 사용하지 않아서 전력소모를 줄이고 데이타 검증을 위한 서버와의 통신비용을 획기적으로 낮출 수 있다. 그림 7은 IR기반 캐시 일관성 유지 기법에서 원하는 데이타가 캐시에 있는 캐

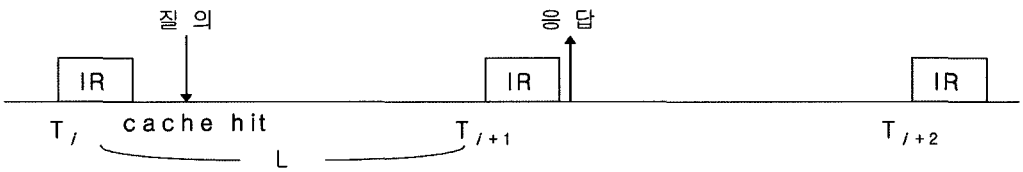


그림 7 IR기반 질의 처리 : 캐시 힛(cache hit) 발생시

시 힛(hit)상황을 그림으로 표현한 것이다. 클라이언트 캐시에 있는 데이타에 대한 질의데이타의 유효성을 확인하기 위해 다음 IR을 수신할 때까지 기다렸다가 수신한 IR에 캐시된 데이타의 이름이 없음(즉, 서버에서 해당 데이타가 갱신되지 않았으므로 클라이언트 캐시의 데이타가 유효함)을 확인하고서야 처리될 수 있다. 이 때문에 질의에서 응답까지 걸리는 데이타 접근시간을 줄이고자 여러 방법들이 제안된 바 있다[8,9].

캐시 미스가 일어난 경우는 캐시 힛보다 응답시간이 매우 길어질 수 있다. 캐시 미스 시, 클라이언트는 서버에 데이타를 요청하고 브로드캐스트 채널을 통하여 해당 데이타가 방송되기를 기다린다. 가장 운이 좋은 경우는 그림 8의 (a)와 같이 현재의 마이너 브로드캐스트 사이클이 끝나고 다음의 마이너 브로드캐스트 사이클에서 요청된 데이타가 방송되는 경우로 IR의 방송주기인 L정도의 시간을 대기하게 되는 경우이다. 이 것은 캐시 hit의 경우와 비슷한 응답 시간을 갖게 된다. 가장 최악의 경우는 그림 8의 (b)와 같이 데이타의 방송 빈도가 가장 낮게 설정되어 여러 개의 마이너 브로드캐스트 사이클 후에 방송되는 경우이다. 이동 클라이언트에서 캐시 미스가 일어난 경우 데이타 접근 시간이 길어지는 문제도 있지만 방송 주기가 매우 낮은 데이타를 수신하기 위해서 오랫동안 브로드캐스트 채널을 듣게 되므로 전력 소모가 커지게 된다. 이것은 자원 제약에 시달리는 이동 클라이언트에게 큰 단점이 될 수 있다.

그림 9는 IR을 사용한 기존의 캐시 유효성 확인 기법에서 장시간의 접속 단절시 캐시의 모든 내용을 실제 유효성에 관계없이 무조건 버리는 상황을 그림으로 표시한 것이다. 그림에서 IR의 윈도우 사이즈는 3이라고 가정하였다. 이것은 IR이 3×L기간 사이의 데이타 갱신에 관한 내용을 모두 포함하므로 2개의 IR을 수신하지 못하더라도 3번째 IR을 통해 그 기간동안의 데이타 갱신 사실을 알 수 있다는 의미이다. 그림 9는 윈도우 사이즈 크기보다 큰 4개의 IR을 수신하지 못하였다. 접속 단절 후 가장 처음 수신한 IR의 타임스탬프 $T_{j+5}$와 마지막으로 수신하였던 타임스탬프 $T_j$의 차이가 3×L보다 크

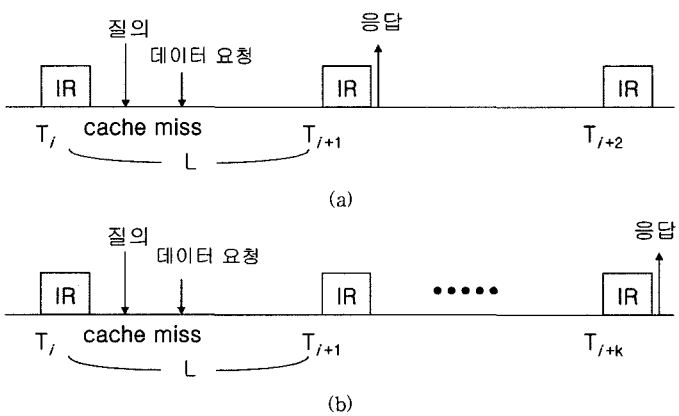


그림 8 IR기반 질의 처리 : 캐시 미스(cache miss) 발생시

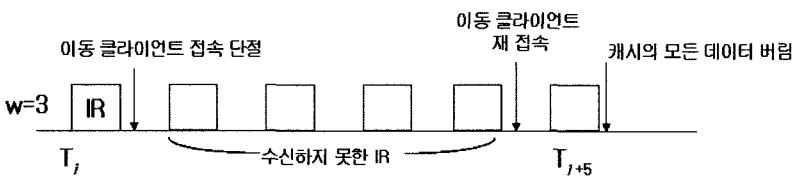


그림 9 긴 접속 단절로 인한 캐시 데이타 무효화

므로, 캐시 내용의 확인 없이 모든 내용을 버리게 된다.

기존의 연구들은 IR이 확인할 수 있는 데이타를 늘이거나 IR대신 전체 데이타베이스에 관한 정보(bit sequence)를 방송하는 등의 방안으로 접속 단절 허용 시간을 최대한으로 하기 위한 것들이 대부분이었다. 또한 캐시의 재사용율에 관한 연구보다는 데이타 접근 시간이나 튜닝시간을 줄이는 것에 중점을 두어 왔다. 하지만 앞의 그림에서 보았듯이 이동클라이언트에서 캐시 미스가 발생되면 데이타 접근 시간과 에너지 소모가 매우 커지므로, 유효한 데이타는 재사용할 수 있도록 캐시 재사용율을 높이는 것이 필요하다.

이 논문에서 제시하는 캐시 일관성 유지 기법은 긴 접속 단절 후 재접속 시에 무조건 캐시의 데이타를 버리는 대신, 캐시의 재사용율을 높이기 위해서 캐시를 버리는 비용과 서버에게 유효성 여부를 질의하는 비용을 비교한다. 이동 클라이언트에 캐시된 데이타들이 만일 갱신율이 아주 낮고 크기가 매우 크며 방송 빈도가 아주 낮다면 이 데이타들은 무조건 버리기 보다는 서버에게 데이타의 유효성을 질의하는 것이 무조건 버리고나서 다시 브로드캐스트채널을 통해 캐시하는 비용보다 더 낮을 수 있다. 반대로 데이타들의 갱신율이 매우 높고 크기가 작으며 방송 빈도 또한 매우 높다면 이 데이타들은 서버에게 데이타의 유효성을 질의하는 것보다는 일단 캐시의 내용을 버리고 브로드캐스트 채널을 통해 캐시하는 것이 시간과 비용측면에서 더 효율적이고 간단하다.

## 4. 캐시 재사용율 향상을 위한 일관성 유지 기법

### 4.1 제안하는 기법의 특징

이 논문에서 기존 클라이언트 중심 캐시 일관성 유지 알고리즘의 문제점들을 해결하기위해 중점적으로 고려한 사항은 다음과 같다.

**① 클라이언트와 서버의 협력**

스테이트리스 서버는 최소한의 정보만을 가지고 이동 클라이언트가 캐시의 데이타 유효성을 정확히 판단하도록 도와주어야 한다. 서버가 캐시 유효/무효 결정을 내려 클라이언트에 전달하기 보다는, 기존의 클라이언트 중심 기법과 마찬가지로 클라이언트가 무효화 보고서를 통해 판단할 수 있도록 무효화 보고서를 구성하여 방송한다. 장시간의 접속 단절 시에는 서버가 클라이언트의 검증 요청에 응답함으로써 유효한 데이타를 무조건 버리게 되는 클라이언트 중심 기법의 단점을 극복할 수 있어야 한다.

**② 데이타 갱신율 고려**

자주 갱신되지 않는 데이타의 재사용율은 다른 데이타 보다 높아야 한다. 데이타의 갱신율을 고려하여, 연결 해제 시간이 길다는 이유로 캐시된 데이타를 버리고 동일한 데이타를 다시 캐시하는 경우를 최대한 배제해야 한다. 클라이언트는 무효화보고서(Invalidation Report)를 수신함으로써 캐싱한 데이타의 갱신율을 추측할 수 있다. 무효화 보고서는 일정 기간동안 서버에서 갱신된 데이타의 이름과 시간이 들어있기 때문에, 무효화 보고서에서 많이 방송된 데이타일수록 갱신율이 높다고 예측할 수 있다. 즉, 캐시된 데이타들 중 갱신율이 높은 데이타들은 장기간의 단절 후 재접속 시 유효하지 않을 확률이 더 높다.

**③ 데이타 방송 빈도 고려**

방송 빈도가 높은 데이타를 다시 캐싱하는 비용은 방송 빈도가 낮은 데이타를 다시 캐싱하는 비용보다 낮다. 따라서 데이타의 캐싱 비용에는 데이타의 방송 빈도가 반영되어야 한다.

**④ 업링크사용 최소화**

클라이언트가 접속 단절 이후 재 연결이 되면, 경우에 따라 일관성의 확인을 위하여 서버에 자신의 캐시 데이타의 유효성을 문의하는 데이타를 전송할 수 있다. 그러나 언제나 캐시의 모든 데이타에 관한 검증 요청을 전송하는 것은 업링크 대역폭과 서버에 많은 오버헤드를 요구하므로 캐시된 데이타의 특성(방송 빈도와 갱신 주기)에 따라 업링크를 사용할 것인지, 업링크를 사용하지 않고 전체 캐시 데이타를 버릴 것인지 효과적으로 판단하여야 한다.

### 4.2 데이타의 분류

이 논문에서 제안하는 캐시 일관성 유지 기법은 접속 단절 시간 외에 데이타의 특성을 반영한다. 데이타의 크기, 갱신율, 방송 빈도 등을 고려하여 긴 접속 단절 후 재접속 시 캐시에서 무조건 버릴 것인지, 서버에게 유효성을 문의할 것인지를 판단한다. 데이타는 방송 빈도(Freq)와 갱신율(Update rate)에 따라 그림 10과 같이 네 가지 유형으로 분류할 수 있다.

먼저 방송 빈도에 따라 핫(hot) 데이타와 콜드(cold) 데이타로 구분한다. 자주 방송되는 주요 데이타나 이동 클라이언트에서 많은 요청이 발생하는 인기 있는 데이타는 방송 빈도가 높은 핫 데이타이다. 이와 반대로 방송 빈도가 매우 낮고 클라이언트로부터 요청이 빈번하지 않는 데이타는 콜드 데이타이다.

데이타들은 또한 갱신율에 의해서 자주 갱신되는 데이타와 자주 갱신되지 않는 데이타로 구분된다. 이 연구에서 데이타의 갱신은 서버에서만 발생한다고 가정하였다. 데이타의 방송 빈도와 갱신율은 각 각 데이타에 대한 읽기요청과 쓰기 요청에 밀접한 관계가 있다.

데이타의 특성에 따라 방송 빈도와 갱신율이 모두 높을 수도 있고, 모두 낮을 수도 있으며, 또는 서로 반대

의 특성을 가질 수 있다.

- **HH형**- 방송 빈도가 높고 갱신율도 높은 데이타. 이러한 유형의 데이타는 긴 시간의 접속 단절 동안 갱신되었을 확률이 높아 재접속 시 유효하지 않을 확률이 크다. 또한 방송 빈도가 높으므로 재 캐싱하기 위해 방송 채널을 수신하면서 기다리는 시간이 짧기 때문에 재 캐싱 비용이 낮다. 데이타를 무조건 버리고 다시 캐싱하는 것이 유리하다.
- **HL형**- 방송빈도는 높지만 갱신율은 낮은 데이타. 자주 변경되지 않는 데이타지만 인기도가 높아 자주 방송되는 데이타다. 이러한 유형의 데이타는 긴 접속 단절 후 재접속 시 데이타를 무조건 버리고 다시 캐싱하는 것이 유리할 수 있지만, 서비스타임(브로드캐스트 채널의 다운링크 대역폭에 대한 데이타 크기의 비율)이 방송 빈도에 비해 아주 높고, 업링크 대역폭이 충분하며, 갱신율이 매우 낮은 경우는 서버에게 유효성을 문의하는 것이 더 유리하고 캐시 재사용율도 높일 수 있다.
- **LH형**- 방송빈도는 낮지만 갱신율이 높은 데이타. 인기도가 낮아 자주 방송되지 않지만 서버에서 매우 빈번하게 갱신되는 경우로 실제로 이러한 유형의 데이타는 매우 드문 경우이므로 모의실험에서는 제외하였다.
- **LL형**- 방송빈도도 낮고 갱신율도 낮은 데이타. 자주 방송되지 않는 데이타로 방송 빈도가 낮으며, 서버에서의 갱신율 또한 낮다. 이러한 데이타는 캐싱하기 위해 오랜 기간동안 브로드캐스트 채널을 듣고 있어야 하며, 갱신율이 낮아 무조건 버리기 보다는 서버에게 유효성을 문의하는 것이 여러 가지 면에서 효율적이다. 특히 LL형이면서 서비스 타임이 큰 데이타(데이타의 크기가 큰 데이타)의 경우 긴 접속 단절 후라도 무조건 캐시에서 버리는 것은 비효율적이다. 데이타가 유효할 확률이 크고 재캐싱 비용은 매우 높기 때문이다.

위와 같은 데이타 분류는 효율적인 비용함수를 도출하기 위해 데이타를 특성에 따라 나누어 본 것으로 실제로 서버나 클라이언트가 데이타를 이 기준에 따라 분류해 놓지는 않는다. 데이타를 기준에 따라 그룹 짓고 그룹에 따라 검증을 하는 DRCI[4]기법과 다르다. 모의실험시에는 가장 높은 방송빈도의 데이타의 Freq값은 20(1회의 주방송 사이클에서 20회 방송), 가장 낮은 방송 빈도의 데이타의 Freq 값은 1로 설정하였고, 갱신율이 높은 데이타의 갱신주기는 3L보다 낮도록, 갱신율이 낮은 데이타의 갱신주기는 5L에서 50L사이의 값으로 랜덤하게 할당하였다.

### 4.3 비용함수

이 절에서는 긴 접속 단절 후 캐시의 데이타를 모두 버릴 것인지, 유효 요청을 보낸 후에 서버의 응답에 따라 선택적으로 버릴 것인지를 판단하기 위한 비용함수를 소개한다. 이동클라이언트는 긴 접속 단절 후 최초로 IR 메시지가 수신된 시점에서 각각의 비용함수를 예측하여 더 낮은 비용이 드는 방법을 선택한다.

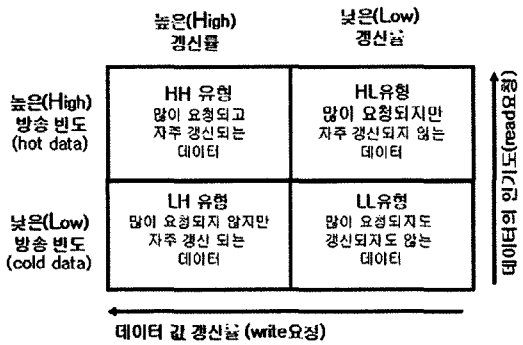


그림 10 방송 빈도와 갱신율에 따른 데이타 분류

#### 4.3.1 캐시의 데이타를 모두 버리는 경우(Purge-All)의 비용함수, *CostPA*

긴 접속 단절 후 무조건 캐시의 데이타를 모두 버리고 다시 캐싱하는데 걸리는 시간을 *CostPA(Purge All)*라고 정의한다. 그림 11에서 보듯이 재접속 후 적어도 한 번의 IR을 수신해야 클라이언트의 접속 단절 시간이 윈도우 크기를 넘어가는 장시간의 접속 단절임을 판명하게 된다. 이 경우, 이동 클라이언트가 캐시에 저장된 데이타를 모두 버리고 브로드캐스트 채널을 수신하여 접속 단절 이전에 캐싱한 데이타를 모두 다시 캐싱하는데 드는 시간을 나타낸 것이다. 캐시의 데이타를 전부 삭제하게 되면 이후 데이타 질의에 대해 캐시 미스가 발생한다. 3.2절에서 보았듯이 캐시 미스 시 데이타 접근시간은 데이타의 방송 빈도에 따라 매우 늦어질 수 있다. 하지만 HH형의 데이타들이 주로 캐싱된 경우 이 비용은 낮아지게 된다.

**정의 4.1.** 브로드캐스트 채널의 하향 대역폭을 $B_{down}$이라고 할 때, 어떤 데이타 $d_x$를 내려 받는데 걸리는 시간을 서비스타임 $Stime(d_x)$라고 정의한다.

$$Stime(d_x) = \frac{|d_x|}{B_{down}(bps)}$$

**정의 4.2.** 접속 단절 전까지 유효했던 캐시 데이타의 집합을 *ValidCache*라고 할 때, 모두 버리고 다시 캐싱하는데 걸리는 시간 *CostPA*는 다음과 같다.

$$CostPA = \left[\frac{MBL}{MIN[Freq_{d_x}|d_x \in ValidCache}\right] + MAX[Stime(d_x)|d_x \ with \min \ Freq]$$

위 식에서 $MBL$은 서버 데이타베이스의 모든 데이타가 적어도 한번은 방송이 되는 주 브로드캐스트 사이클

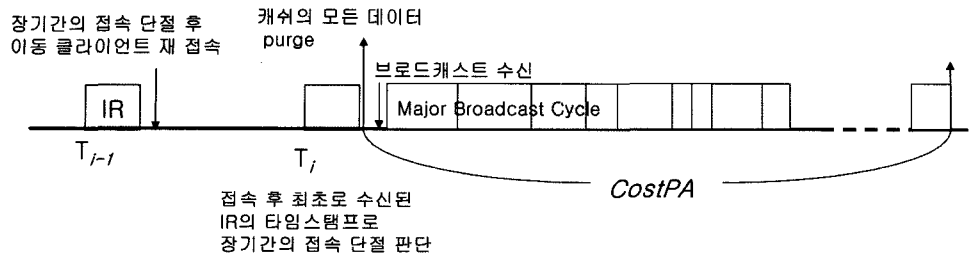


그림 11 캐시 내 모든 데이타 삭제 비용

의 전체 길이(Major Broadcast-cycle Length)이다. MBL을 캐싱 되어 있던 데이타 중 가장 낮은 방송 빈도 값으로 나누면 이 데이타를 수신하기위한 대기 시간을 구할 수 있다. 여기에 이 데이타들을 캐싱하기 위해 내려받는 시간이 필요하므로 최소 방송 빈도의 데이타 중 가장 큰 서비스 타임 값을 더해준다. 여기서 가장 방송 빈도가 낮은 데이타만을 고려한 이유는 방송빈도가 높은 데이타는 낮은 방송빈도의 데이타를 수신하기 위해 기다리는 동안 수신할 수 있기 때문이다.

4.3.2 캐시 데이타를 선택적으로 버리는 경우(Selective Purge)의 비용함수, *CostSP*

이동 클라이언트 캐시에 저장된 데이타에 대해 서버에 검증 요청을 보내고 서버로부터 유효성이 확인되지 않은 데이타만을 선택적으로 버린 후 이들 데이타만을 다시 캐싱 하는데 드는 시간 *CostSP*라고 정의한다. 즉, 이동 클라이언트가 업링크 채널을 사용하여 서버에 캐시 데이타의 검증 요청을 한 후 무효 판정된 데이타만 선택적으로 삭제하여 삭제한 데이타들을 다시 브로드캐스트 채널에서 수신하는데 걸리는 시간이다(그림 12). 서버에 검증 요청을 보내기 위해 업링크를 사용하므로 업링크 메시지 크기와 업링크 대역폭의 크기가 중요한 변수가 되고, LL형의 데이타들이 주로 캐싱된 경우 이 비용이 낮아진다.

**정의 4.3.** $T_j$를 접속 재개 후 가장 처음 수신된 IR의 타임스탬프라고 할 때, IR의 윈도우 사이즈를 넘는 장시간 접속 단절 동안 무효화 되었을 가능성이 있는 데이타의 집합 P(Set Purge)는 다음과 같이 정의한다.

$$P=\left\{ d_x \mid \frac{L}{IRRate_{d_x}} \leq (T_j - last\,TS),\ \text{for}\ \ d_x \in ValidCache \right\}$$

정의 4.3은 캐시 데이타 중 접속 단절 기간동안 갱신되었을 확률이 큰 데이타의 집합을 $P$라고 정의한 것이다. 각각의 데이타마다 클라이언트가 수신한 전체 무효화 보고서의 갯수와 해당 데이타가 포함된 무효화 보고서 갯수의 비율 *IRRate*를 가지고 있다. 만일 전체 수신한 IR이 100개이고 이 중 50개에 $d_x$가 포함되어 있다면 $d_x$는 평균 2L기간 동안 적어도 한 번은 갱신이 된 것이다. 만일 수신한 100개의 IR중에서 2개에만 $d_y$가 포함되어 있다면 $d_y$의 갱신 주기는 평균 50L이 된다. 접속 단절 시간($T_j - last\,TS$)이 10L이라면 이 기간 동안 $d_x$가 갱신 되었을 확률이 $d_y$가 갱신되었을 확률보다 매우 크고, $d_x$의 갱신 주기가 접속단절 시간보다 적으므로 집합 P에 포함시킨다. 그러나 $d_y$의 평균 갱신 주기50L은 접속 단절 시간 10L 보다 크므로 집합P에 포함시키지 않는다.

**정의 4.4.** 브로드캐스트 채널의 상향 채널 대역폭을 $B_{up}$, 데이타 아이디의 크기를 $d_{id}$, 타임스탬프 LastTS의 크기를 $T_{id}$라고 할 때 클라이언트가 캐시 데이타 유효성 확인을 위해 업링크 메시지를 보내는 VRtime (Validation Request Time)을 다음과 같이 정의한다.

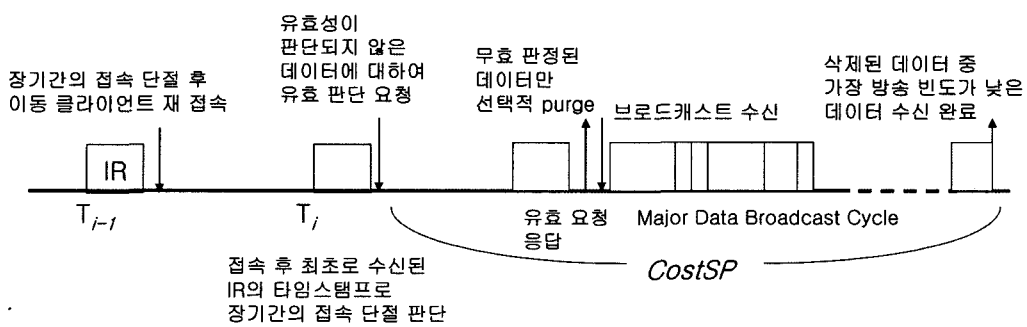


그림 12 캐시 데이타의 선택적 버림 비용

$$VRtime = \frac{(|ValidCache - cached\ \ data\ d_x \notin IR_{T_j} - P|) \times |d_{id}| + |T_{id}|}{B_{up}}$$

*VRTime*은 클라이언트에서 업링크 채널을 통해 데이타 유효확인 요청을 서버에 보내는 시간으로 유효 요청하는 데이타의 수와 업링크 채널의 대역폭에 의해 결정된다. 클라이언트는 캐시에 있는 데이타 중에서 접속 단절 후 처음 수신된 $IR_{T_j}$에 포함되어 있는 데이타를 제외하고, 갱신 가능성이 있는 데이타도 제외한 후 접속 단절 전까지 유효했던 나머지 데이타에 대해 그림 13과 같이 데이타 이름과 유효성을 보장한 마지막 타임스탬프로 구성된 검증 요청 메시지를 보낸다.

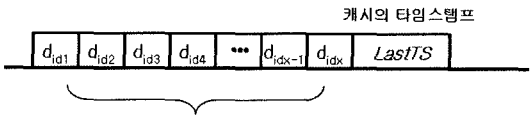


그림 13 검증 요청 메시지

**정의 4.5.** 서버에 검증 요청을 하여 무효 데이타만 선택적으로 삭제한 후 이들을 재캐싱 하는 비용 *CostSP*는 다음과 같이 정의한다.

$$CostSP = VRtime + L + \frac{MBL}{MIN[Freq_{d_x} | d_x \in P]} + MAX[Stime(d_x) | d_x with\ \ the\ \ smallest\ Freq, \text{ for } d_x \in P]$$

클라이언트의 데이타 유효 확인 요청메시지에 대한 서버의 응답은 다음 IR방송 주기 L이후에 방송된다고 가정한다. 실제로는 그 이후에 방송되기도 하지만 기존의 다른 연구들에서도 서버의 응답 예상 시간을 IR방송 주기 직후로 예측하는 경우가 많다[9-11]. 서버는 클라이언트의 검증 요청 메시지의 데이타 이름(Data *id*)목록과 자신이 유지하고 있는 데이타 갱신 목록을 조사하여 데이타들의 갱신 시간을 $(d_x, t_x)$짝으로 다음 IR메시지 직후에 방송한다. 따라서 위의 정의에서 L을 더하는 것은 클라이언트의 검증 요청에 대한 응답시간이다.

*CostSP*의 세 번째와 네 번째 항목은 선택적으로 삭제한 데이타를 다시 캐싱하는데 걸리는 시간이다. 집합 P는 갱신 예측 시간이 접속 단절 시간이 보다 작은 데이타의 집합이다. 즉, 접속 단절 시간동안 갱신되었을 확률이 크므로 캐시에서 삭제한 후 다시 캐싱하기 위한 시간이다. *CostSP*는 단지 예측 값으로, 집합 P에 속하지 않은 데이타로 서버에게 검증 요청을 보낸 경우에 무효판정을 받은 데이타가 있을 수 있다. 이 경우, 해당 데이타를 캐싱하기 위해 방송 채널을 듣는 시간이 더 길어질 수 있다.

서버는 동시에 여러 개의 클라이언트로부터 데이타 확인 요청을 받을 수 있으며, 하나의 데이타가 동시에 여러 개의 확인 요청 목록에 포함된 경우에도 서버는 한 번만 방송되면 된다. 정의 4.2와 4.5의 비용은 단지 예측 값으로 각각의 경우(캐시를 모두 삭제하고 새로 수신하는 경우와 선택적으로 삭제하고 삭제한 데이타만 수신하는 경우)의 비용을 비교하는 척도로 사용된다. 다음 장의 알고리즘을 통해 이 예측 값이 어떻게 사용되는지 살펴보도록 한다.

### 4.4 재사용율 향상을 위한 캐시 일관성 유지 알고리즘

LL형의 데이타는 갱신 확률도 낮고 방송 빈도도 낮아 접속 단절시간동안 갱신되었을 확률이 낮고, 다시 수신하는데 시간이 오래 걸린다. 반면에 HH형의 데이타는 접속 단절 시간 동안 갱신되었을 확률이 높고, 방송 빈도도 높으므로 서버에게 검증을 요청하기보다는 무조건 캐시에서 버리고 브로드캐스트 채널을 수신하여 다시 캐싱 하는 것이 바람직하다. 그림 14는 이러한 특성을 반영하면서 캐시 재사용율을 높일 수 있도록 제안하는 이동 클라이언트의 캐시 일관성 유지 알고리즘이다.

이 알고리즘은 클라이언트의 접속 단절 시간이 유효 검증 윈도우의 크기를 벗어나지 않으면, 기존의 다른 방법과 마찬가지로 IR에 기반 하여 캐시의 일관성을 유지한다. 윈도우 크기보다 큰 접속 단절의 경우 기존의 방법들은 무조건 캐시의 데이타를 유효하지 않은 것으로 보고 버리는데 비해, 본 기법의 알고리즘은 무조건 버리는 경우와 서버에게 검증요청을 하여 선택적인 버림을 하는 경우의 캐시 유지비용의 예측 값을 계산한다. 즉 4장의 비용함수 *CostPA*(무조건 버리는 경우)와 *CostSP*(캐시 데이타의 선택적 버림)를 예측하여 CostPA가 더 작은 경우는 기존의 방법과 마찬가지로 캐시 데이타를 전부 버리고 브로드캐스트 채널을 통하여 새로 데이타를 수신한다. 반대로 *CostSP*의 비용이 더 작은 경우는 캐시의 데이타를 무조건 버리는 것이 아니라 갱신 확률이 높은 P에 속한 데이타를 버리고 그 외의 데이타에 대해 서버에 검증 요청을 하여 유효하지 않은 것으로 판명된 데이타만을 캐시에서 삭제한다. 캐시된 데이타의 대부분이 LL형인 경우 *CostPA*가 높고 *CostSP*가 낮아지게 되므로 서버에게 검증 요청을 보내게 되고, 반면에 캐시된 데이타의 대부분이 HH형인 경우 *CostPA*가 더 낮아지므로 기존의 방법과 마찬가지로 캐시의 데이타를 전부 삭제한다.

## 5. 성능 평가

본 논문에서 제안한 캐시 일관성 유지기법의 성능 평가를 위해 주로 캐시 재사용율 측면에서 DRCI기법[4]과 비교하여, CSIM[12]을 사용하여 모의 실험하였다. DRCI기법은 대표적인 클라이언트 중심 캐시 검증(client-only cache invalidation) 방법으로 서버의 데이

```
/* IR: 무효화 보고서 (Invalidation Report)                    T_j : 무효화 보고서의 타임스탬프
   LastTS : 가장 최근에 받은 무효화 보고서 타임스탬프          L : 무효화 보고서의 주기,
   w : 무효화 보고서 윈도우 크기(window size)                 Freq_x : 데이터 d_x의 방송 빈도(MBL안에서의 방송 횟수)
   r_x: 데이터 d_x의 갱신 예측을 위한 IR 방송율(d_x가 포함된 IR수/수신된 전체 IR수)
   MBL: 주 방송 사이클의 길이 (Major Broadcast Cycle Length)
   ST(d_x) : 데이터 d_x를 내려 받는데 걸리는 시간
   VRtime : 서버에게 검증요청(Validation Report)을 전송하는데 걸리는 시간  */

   while (true) // 브로드캐스트 채널 수신
      Upon receiving IR of TimeStamp T_j
            if ( T_j - LastTS ≤ w × L ) {      //윈도우를 넘지 않는 짧은 시간의 단절
                  DRCI, BGI와 같은 기존의 캐시 검증 기법을 사용하여 캐시 관리;
                  LastTS = T;
             }
            else {                              //윈도우를 초과하는 긴 접속 단절
                  for all data listed in IR_{T_j} (Invalidation Report of T_j) {
                     if (data d_x is in the cache){
                        purge d_x from the cache;
                     }
                  }
                  // Calculate CostPA- 4장의 정의에 따라 캐시 삭제 비용 예측
                  CostPA = MBL / MIN[Freq_x | d_x ∈ cache] + MAX[ST(d_y | d_y with the smallest Freq_y];
                  // Calculate CostSP- 4장의 정의에 따라 선택적 버림 비용 예측
                  L / r_x < T_j − LastTS인 d_x만을 선택하여 집합 P구성; // 갱신 확률이 높은 데이터만 P에 선별
                  CostSP = VRtime + L + MBL / MIN[Freq_x | d_x ∈ P] + MAX[ST(d_y | d_y ∈ P, Freq_y is the smallest];
                  // 비용 비교
                   if (CostPA ≤ CostSP)
                        purge the entire cached data objects;     // 전체 캐시 데이터를 삭제
                   else
                        {
                           P에 속하는 데이터는 캐시에서 삭제;
                           나머지 데이터에 대하여 VR(Validation Request)메시지 구성하여 서버에게 전송;
                            // 서버에게 검증을 요청
                        }
            }
```

그림 14 이동클라이언트의 캐시 일관성 유지 알고리즘

타를 그룹으로 묶어 그룹별로 검증을 한다. 실험을 통해 데이타 접근 시간 면에서 우수한 성능이 증명된 바 있지만 에너지 소비측면에서는 중간 수준의 성능을 보이고 있고 긴 접속 단절시의 캐시 재사용율에 관한 성능 평가는 이루어 진 바 없다.

DRCI도 다른 IR기반 기법과 마찬가지로 접속단절 시간이 윈도우크기보다 크면 해당 데이타가 속한 그룹의 모든 데이타를 캐시에서 삭제한다. DRCI는 캐시 데이타를 그룹별로 삭제하므로 전체 캐시를 무조건 삭제하는 다른 클라이언트 중심 기법보다는 캐시의 재사용율이 높지만, 그룹별 IR을 별도로 관리하고 방송해야 하는 오버헤드와 그룹을 구성하는 방법에 따라 성능의 차이가

표 2 실험 변수 및 값

| 정의 | 설명 | 기 본 값 |
|---|---|---|
| 공통 변수 | | |
| $D$ | 서버 데이타의 수 | *100,000* |
| $B_{up}$ | 상향채널(uplink) 대역폭 | 64Kbps, 128K, 256K, 1M(1024K) |
| $B_{down}$ | 하향채널(downlink) 대역폭 | *192K, 256K, 512K, 1M, 2M, 3M* |
| $d_{id}$ | 데이타 아이디의 크기 | *32bits* |
| $T_{id}$ | 타임스탬프의 크기 | *64bits* |
| $L$ | 무효화 보고서의 주기 (L) | *60sec* |
| $v$ | 연결 단절 시간(missed IR) | *60sec(1), 120(2),.....3000sec(50)* |
| $Freq$ | 방송 빈도 (Hot: Cold) | *20:1* |
| $\|d_x\|$ | 데이타의 크기 | *1K ~ 1M* |
| $\alpha:\beta$ | hot data : cold data ($\alpha+\beta=100$) | *80:20, 50:50, 20:80* |
| $\alpha_h:\alpha_L$ | 갱신율에 따른 hot data 구성- HH:HL | *80:20, 50:50, 20:80* |
| $\lambda_h$ | HH데이타의 갱신주기 | *1sec ~180sec (<= 3L)* |
| $\lambda_l$ | HL. LL 데이타의 갱신주기 | *300sec ~ 3005sec* |
| $w$ | 윈도우 크기 | *3* |
| DRCI(Dual Report Approach)의 변수 | | |
| $G$ | 그룹 크기 | *100* |
| $N$ | 그룹 갯수 | *1000 ($\approx D/G$)* |
| $G_{id}$ | 그룹 아이디 크기 | *16 bits* |

많이 난다는 단점이 있다. 본 논문이 제안하는 기법이 데이타들을 특징에 따라 그룹으로 나누므로, 이를 DRCI에 그대로 적용하여 실험하였다. 표 2는 실험에서 사용한 파라미터 값을 정리한 것이다.

### 5.1 상향 채널 대역폭과 상향 채널 오버헤드(VRTime)

이 실험은 본 논문의 캐시 검증 기법에서 서버에게 검증을 요청하기위해 사용하는 상향채널의 오버헤드를 알아보기 위한 것이다. DRCI와 같은 클라이언트 중심의 캐시 검증 기법은 서버와의 통신을 배재하고 긴 접속 단절 시에는 캐시의 내용을 버리므로 재사용율은 낮은 대신 상향채널 오버헤드는 거의 없다. 그러나 본 논문이 제시하는 클라이언트 서버 협력의 캐시 검증 기법은 캐시 재사용율을 높이기 위해, 캐싱된 데이타의 특성에 따라 검증 요청 메시지를 서버에 보내기도 한다. 이 경우 발생하는 오버헤드를 예측하기 위해 검증 요청 메시지를 보내는 시간 *VRTime*을 계산하였다. 상향채널 대역폭의 값은 최저 64K, 128K, 256K, 1Mbps로 각각 설정하였다. 그림 15의 실험 결과가 보여주듯이 이동클라이언트에서 서버로 검증 요청할 데이타의 갯수가 증가함에 따라 *VRtime*이 증가하지만, 그 기울기가 별로 크지 않고, 더우기 업링크의 대역폭이 커지면 증가 기울기가 매우 작아지므로 *VRtime*이 심각한 오버헤드를 초래하지 않을 것임을 알 수 있다.

### 5.2 접속 단절 시간과 캐시 재사용율

접속 단절 시간에 따른 캐시 재사용율을 캐시된 데이타의 방송빈도에 따라 비교하였다. 캐싱된 데이타의

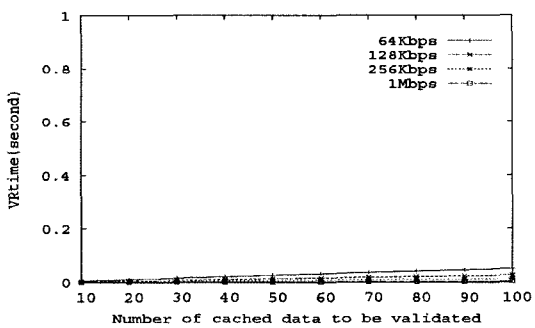


그림 15 캐시 데이타 수와 상향채널 오버헤드 (Validation Request Time)

80%, 50%, 20%가 자주 방송되는 핫 데이타인 경우에 대해 본 논문이 제안하는 기법의 캐시 재사용율을 DRCI 기법과 비교한 것이다. IR을 사용한 기본적인 타임스탬프 기법은 단절 시간이 윈도우 크기를 넘어서면 무조건 캐시의 내용을 버리므로 재사용율이 0%가 된다. DRCI기법도 접속 단절시간이 그룹윈도우 크기 6을 넘어서면 재사용율이 0%가 된다. 그림 16에서 보듯이 본 논문이 제안하는 기법은 기존 IR기반 기법 BT나 DRCI에 비해 매우 향상된 재사용율을 유지한다. 특히 캐싱된 데이타의 갱신율이 낮은 경우 재사용율이 가장 높다.

### 5.3 데이타 갱신율과 캐시 재사용율

이 실험은 데이타 갱신율에 따른 캐시 재사용율을 측정해 본 것이다. 서버의 전체 데이타 중 자주 갱신되는 (갱신주기 < L) HH 유형에 해당하는 데이타의 비율에

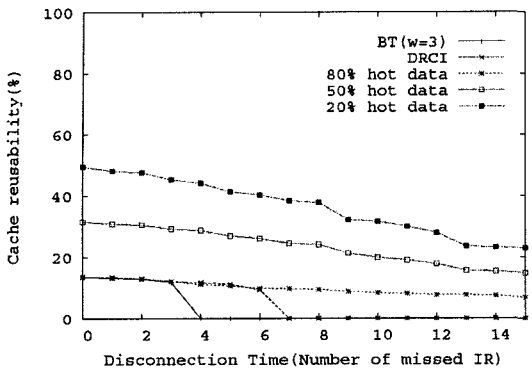


그림 16 접속 단절 시간과 캐시 재사용율

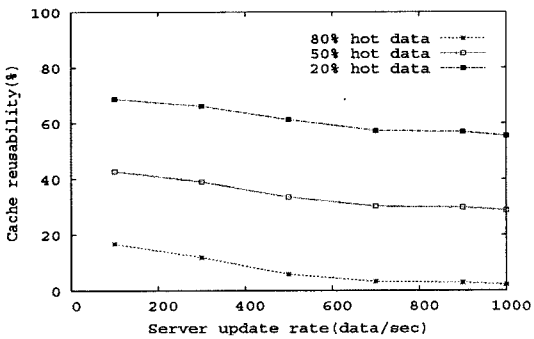


그림 17 서버 데이타 갱신율과 캐시 재사용율

따른 캐시 재사용율을 알아보았다. 데이타의 실제 갱신율과 클라이언트가 캐싱한 데이타의 갱신율을 추측하기 위한 *IRRate*에는 차이가 있다. 비록 *IRRate* 값이 실제 갱신률보다 낮은 값을 가지지만, IR 방송 주기보다 갱신 주기가 짧은 데이타는 모두 재캐싱해야 될 데이타(집합 P)로 분류된다. 그림 17에서 보듯이, 제안하는 알고리즘은 캐시 데이타의 대부분(80%)이 자주 갱신되지 않는 콜드 데이타인 경우 서버에서 데이타의 갱신율에 상관없이 경우 높은 캐시 재사용율을 보임을 알 수 있다.

### 5.4 캐시된 데이타 평균 크기와 재사용율

데이타의 접속 단절시간이 윈도우 크기보다 클 때, 캐시된 데이타의 평균 크기를 1K에서 1M까지 변화시키면서 재사용율을 측정하였다. 이 때 데이타의 갱신 주기는 HH데이타인 경우 L보다 작은 값을 갖도록 하였고(평균 1/2L), HL형의 데이타인 경우 300초(5L)에서 3005초(>50L)의 값을 갖도록 하였다. 그림 18은 데이타의 크기와 갱신빈도사이에 아무런 상관관계가 없을 때의 실험이고 그림 19는 데이타의 크기와 갱신주기가 비례하도록 설정한 실험의 결과이다. 즉, 두 번째 실험은 데이타의 크기가 클수록 자주 갱신되지 않고, 자주 갱신되는 데이타일수록 그 크기가 작다고 가정한 것이다(데이타

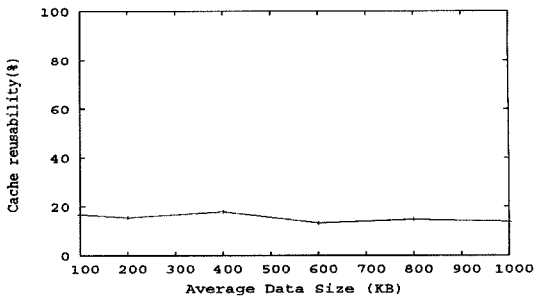


그림 18 평균 데이타 크기와 재사용율
(무작위 갱신 주기)

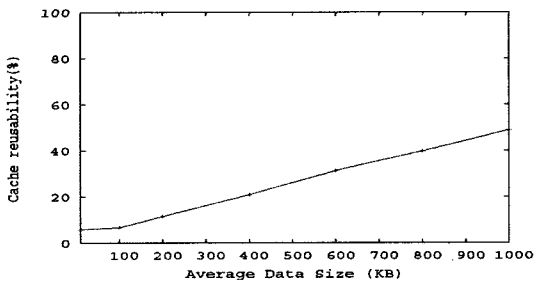


그림 19 평균 데이타 크기와 재사용율
(갱신주기와 데이타 크기 반비례)

의 크기가 커질수록 갱신주기가 커지도록 설정하였다.) 실험 결과에서 보듯이 데이타의 크기와 갱신 주기 사이에 아무런 연관성이 없는 경우, 캐시된 데이타의 평균 크기가 재사용율에 별다른 영향을 미치지 않지만, 데이타의 크기와 갱신 주기가 비례하는 경우에는(즉, 데이타의 크기와 갱신율이 반비례하는 경우) 캐싱된 데이타의 크기가 크면 재사용율이 매우 높아진다.

### 5.5 대역폭에 따른 캐시 재사용율

상향 채널의 대역폭과 하향 채널의 대역폭 비율에 따른 캐시 재사용율을 측정하였다. 현재 대부분의 무선 랜이나 와이브로와 같은 무선통신기술에서 상향 채널의 대역폭은 하향 채널 대역폭의 최대 3분의 1 수준이다. 이 실험에서는 상향 채널과 하향 채널의 대역폭 비율을 1:1, 1:3, 1:5로 설정하여, 긴 접속 단절시(missed IR > 10)의 캐시 재사용율을 측정해 보았다. 하향 채널의 대역폭이 상대적으로 축소될수록 재캐싱 비용이 높아지므로 캐시 데이타의 선택적 삭제가 선호되어 캐시 재사용율이 높아지게 된다. 반대로 하향 대역폭이 상대적으로 큰 경우 캐시 전체를 삭제하고 재캐시하는 비용이 낮아지므로 전체 삭제가 선호된다. 그림 20에서 보듯이 캐시의 재사용율은 상향 대역폭과 하향대역폭의 비율에 영향을 받는다.

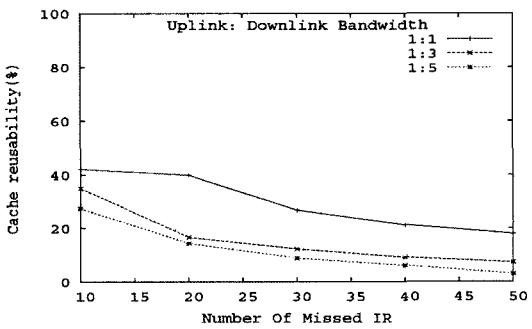


그림 20 긴 접속 단절 시 대역폭비율에 따른 캐시 재사용율

## 6. 결론 및 향후 연구 방향

이 논문은 브로드캐스팅 타임스탬프와 무효화 보고서를 기반으로 한 클라이언트 중심 캐시 유효성 확인 기법이 가지고 있는 문제점-장기간의 접속 단절시 캐시 재사용율이 급감한다는 문제점을 개선하는 클라이언트-서버 캐시 검사 기법을 제안하였다. 이 논문이 제안하는 캐시 일관성 유지 기법은 이동 클라이언트의 연결 해제 시간뿐 아니라 캐시된 데이타의 갱신 주기, 방송 빈도 등을 함께 고려하였다. 이동 클라이언트는 장시간의 접속 단절 후 무조건 캐시 데이타를 버리는 것이 아니라 캐시된 데이타의 방송 빈도와 갱신 주기를 고려하여 삭제 후 다시 캐싱하는 비용과 서버에게 검증 요청하는 비용을 예측하여 비교한다. 캐시의 모든 데이타를 버릴 것인지 아니면 서버에게 검증 요청을 보낼 것인지 비용면에서 유리하도록 결정하므로 캐시된 데이타의 특징에 따라 캐시의 재사용율을 높일 수 있다. 이때 발생하는 서버와의 통신비용은 심각한 오버헤드가 되지 않고, 캐시된 데이타의 특성에 따라 캐시 재사용율을 매우 높일 수 있음을 모의실험을 통해 보였다.

결론적으로 이 논문이 제시하는 캐시 일관성 유지 기법은 장시간의 접속 단절이 발생한 경우라도, 캐시된 데이타의 갱신이 거의 일어나지 않는다면 높은 재사용율을 보장하여 클라이언트의 전력 소모를 줄이고, 이 데이타들의 방송 빈도가 낮은 경우에는 데이타 접근 시간도 단축 한다. 이 기법은 서버와의 통신 오버헤드가 서버중심 기법처럼 높지 않으므로 여전히 확장성(scalability) 측면도 보장한다.

향후 연구 방향은 다음과 같다. 첫째, 이 논문은 주로 캐시 재사용율에 초점을 맞추어 실험이 진행되었다. 클라이언트의 질의 도착율에 따른 데이타 접근 지연 시간 등 보다 다양한 실험 환경에서 데이타 접근 시간을 중심으로 좀 더 많은 실험을 진행할 계획이다. 데이타 접근 시간외에도 튜닝 시간, 스트레치(stretch: 데이타 접근 지연 시간과 서비스 시간의 비율)등에 관한 실험도 필요하다. 데이타의 갱신 주기와 접근 지연 시간, 클라이언트의 질의 도착 시간에 따른 캐시 재사용율과 데이타 접근 시간의 총체적인 상관관계를 포함할 수 있는 실험이 수행될 예정이다. 캐시 재사용율을 높임으로써 궁극적으로 이동 클라이언트의 데이타 접근 시간, 스트레치, 이동 클라이언트의 전력 소모 등이 개선되리라 기대되므로 이때 야기되는 클라이언트의 통신 오버헤드는 충분히 보상을 받을 수 있다는 것을 좀 더 다양한 실험을 통해 보이고자 한다.

둘째, 이 논문에서는 서버 데이타의 갱신 주기가 일정하다고 가정하였다. 즉 클라이언트가 캐시 데이타의 갱신 주기를 IR에 의해 예측하는 것이 가능하다고 가정하였다. 하지만 데이타의 갱신 주기가 동적으로 변하는 경우 실제 갱신율과 클라이언트가 예측하는 갱신율에 차이가 많이 나게 되고, 비용 함수가 갱신 주기를 반영하는 것이 어려워 진다. 이러한 경우를 대체할 수 있는 방안에 관한 연구가 필요하다.

마지막으로 이 논문의 실험들은 캐시 일관성에 집중하기 위해 이동 클라이언트에서 캐시 리플레이스먼트[11]가 일어난 경우를 배제하였다. 캐시 리플레이스먼트 정책에 따라 캐시 재사용율이 영향을 받고, 이동 클라이언트의 경우 방송 빈도와 갱신 주기가 리플레이스먼트 정책에 반영되어야 할 것이므로, 본 연구를 확장하여 캐시 리플레이스먼트 정책과도 연계할 계획이다.

## 참 고 문 헌

[1] D. Barbara, and T. Imielinski, "Sleepers and workaholics: caching strategies in mobile environments," *ACM-SIGMOD*, pp.1-12, June 1994.

[2] J. Cai and K. Tan, "Energy-efficient selective cache invalidation," *Wireless Networks*, vol.5, pp. 489-502, 1999.

[3] J. Jing, A. Elmagarmid, A. Helal and R. Alonso, "Bit-Sequences : An Adaptive Cache Invalidation Method in Mobile Client/Server Environments," *ACM Mobile Networks and applications* Vol. 2, pp.115-127, 1997.

[4] K. Tan, J. Cai and B. Ooi, "An Evaluation of Cache Invalidation Strategies in Wireless Environments," *IEEE Transactions on parallel and distributed systems*, vol. 12, no. 8, pp.789 - 807, August 2001.

[5] S. Yi, W. Song, S. Jung, "A Cost Effective Cache Consistency Method for Mobile Clients in Wireless Environment," *Data base Systems for Advanced Applications*, pp.908-915, 2004.

[6] J. Cai, K. Tan, "Energy-efficient selective cache

invalidation," *Wireless Networks*, No.5, pp. 489-502, 1999.

[7] S. Yi, H. Shin, S. Jung, "Enhanced Cost Effective Cache Invalidation for Mobile Clients in Stateless Server Environments," *Embedded and Ubiquitous Computing*, pp.387-397, 2004.

[8] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," *IEEE Trans. on Knowledge and data Engineering*, vol.15, No.5, pp. 1251-1265, 2003.

[9] M. Yeung and Y. Kwok, "Wireless Cache Invalidation Schemes with Link Adaptation and Downlink Traffic," IEEE Transactions on mobile computing, vol.4, no.1 68-83, January/February 2005.

[10] A. Kahol, S. Khurana, and S. Gupta, "A Strategy to Manage Cache Consistency in a Disconnected Distributed Environment," *IEEE Transactions on parallel and Distributed systems*, vol. 12, no. 7, July 2001.

[11] J. Xu, Q. Hu, W. Lee, "Performance of Evaluation of an Optimal Cache Replacement Policy for Wireless Data Dissemination," IEEE Transactions on knowledge and data engineering, Vol.16, No.1, 125-139, January 2004.

[12] K. Watkins, Discrete Event Simulation in C. McGraw-Hill, 1993.

이 송 이

1988년 이화여자대학교 전자계산학 학사 1990년 M.S. Michigan State University. 1997년 서울대학교 컴퓨터 공학과 박사 1997년~1998 University of Wisconsin-Madison, Post-Doc. 2000년~2003년 성신여자대학교 컴퓨터 정보학부 강의전담 교수. 2004년~현재 서울대학교 BK21정보기술사업단 계약 교수