무선 환경에서의 이동 클라이언트를 위한 효율적인 캐시 일관성 유지 방안

송 원민[0] 정 성원
서 강 대 학 교 컴 퓨 터 학 과
sjohn@mclab.sogang.ac.kr jungsung@ccs.sogang.ac.kr

An Efficient Cache Consistency Method for Mobile Clients in Wireless Environment

Wonmin Song[0] Sungwon Jung
Dept. of Computer Science, Sogang University

요 약

최근 이동 컴퓨팅(Mobile Computing)환경에서의 잦은 접속 단절로 인한 클라이언트 캐시 일관성(Consistency) 문제를 해결하기 위한 방법으로 클라이언트 캐시 유지에 대한 연구가 진행되고 있다. 이러한 캐시 유지 방법의 한 분야로서, 캐시 데이터의 일관성 문제를 해결하기 위한 무효화 보고(Invalidation Report, IR)에 대한 연구가 진행 중이다. 그러나 기존의 무효화 보고는 시간 의존적인 정보로써 전송되기 이전 일정 시간동안의 서버 업데이트 정보만을 포함한다. 따라서 서버의 업데이트 정도와 무관하게 일정 시간 이상의 클라이언트 접속 단절에 대하여 클라이언트의 캐시 일관성을 유지하지 못한다. 따라서 본 논문에서는 시간 의존적 정보를 가지는 무효화 보고와 서버의 업데이트 정도에 따른 무효화 보고를 함께 사용하여 클라이언트의 접속 단절 시간과는 무관하게 서버의 업데이트 정도에 따라 캐시 일관성을 유지할 수 있는 효율적인 기법을 제안한다.

1. 서 론

이동 컴퓨팅 환경에서는 대역폭의 비대칭성으로, 클라이언트가 특정 데이터에 접근을 원할 때마다 서버에 요청하는 것은 심각한 대역폭의 부족과, 클라이언트 내의 질의 처리 시간의 증가를 초래한다. 따라서, 클라이언트는 자주 사용되는 데이터를 캐시에 저장하여 이러한 문제를 해결하고, 서버는 무효화 보고 (Invalidation Report, IR)를 클라이언트에 전송하여 데이터 일관성(Consistency)을 보장한다. 그러나 무선 환경에서 클라이언트는 서버와의 불안정한 연결로 인하여 잦은 접속 단절이 발생하며, 이로 인하여 서버에서 전송되는 IR을 받지 못하면 캐시의 일관성을 유지할 수 없으므로 심각한 효율성의 저하가 발생한다.

이에 관하여 Bit Sequence (BS) [1], Broadcasting Timestamps (BT) [2], *Adaptive Invalidation Report with Adjusting Window* (AAW) [3] 등의 방법이 제안되었다. BS 기법은 서버 업데이트 정보를 계층화된 비트열로 전송하였으며, BT 기법은 IR의 정보의 범위를 증가시켰고, AAW 기법은 동적으로 BS 기법과 TS 기법의 사용을 결정한다. 그러나 이러한 방법들은 수신하지 못한 IR의 수에 대한 제한이 있었다. 이는 서버에서의 업데이트 정도와 무관하여 서버에서 업데이트가 자주 발생하지 않은 경우 클라이언트의 캐시 대부분을 유지할 수 있음에도 불구하고 모든 데이터를 버려야만 하는 문제점을 가지고 있었다.

따라서 본 논문에서는 위의 문제점을 반영하여 서버의 접속 단절 시간과 서버의 업데이트 정도를 함께 고려하여 캐시 일관성을 유지할 수 있는 기법을 제안한다. 서버는 클라이언트 캐시의 데이터를 알지 못하나 클라이언트 캐시 데이터를 예측하여 업데이트 여부를 확인함으로써 업데이트가 적은 경우라면 클라이언트 접속 단절 동안의 업데이트 정보를 전송하며, 일정 수준 이상의 업데이트가 발생하여 캐시 유지비용이 큰 경우에는 모든 데이터를 버리도록 하여 클라이언트 캐시의 일관성을 유지할 수 있도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 제안하는 IR을 종류별로 분류하였고, 3장에서는 이러한 IR을 이용한 클라이언트의 작업 모델을, 4장에서는 IR을 전송하기 위한 서버의 판단 방법을 설명한다. 5장에서는 제안된 기법에 대한 IR의 전송 비율을 측정하였다.

2. 무효화 보고의 종류

캐시 일관성 유지를 위한 한가지 방법으로 IR에 대한 연구가 진행 중이다. IR이란 주기적, 혹은 비주기적으로 이전 IR이 전송된 이후에 대한 서버 업데이트 정보를 브로드캐스트 하여 클라이언트가 캐시 데이터의 일관성을 유지할 수 있도록 하는 방법이다. 그러나 본 논문에서는 이러한 일반적인 IR이 아닌 서로 다른 세 종류의 IR을 제안한다.

Common Invalidation Report (CIR)

L의 주기를 가지고 브로드캐스트 되는 IR로서 서버에서 전송되는 시점의 timestamp와 전송되기 이전 L 간격의 시간동안 서버에서 업데이트된 데이터의 이름(Name)들을 저장하고 있다.

CIR의 내용 : <Timestamp, Name>

Requested Invalidation Report (RIR)

하나 이상의 CIR을 수신하지 못한 클라이언트가 서버에 캐시 데이터의 일관성 확인 요청(Validation Request : VR)을 할 때 전송되는 IR로서, 클라이언트의 접속 단절 시간동안 업데이트된 데이터가 적어 클라이언트 캐시 데이터의 많은 부분을 유지할 수 있다고 판단될 때 전송되며, 서버에서 전송되는 시점의 timestamp와 클라이언트의 접속 단절 시간동안 업데이트된 데이터의 이름(Name)들을 저장하고 있다.

RIR의 내용 : <Timestamp, Name>

Drop Message (DM)

하나 이상의 CIR을 수신하지 못한 클라이언트가 VR을 전송할 때 전송되는 IR로서, 접속 단절 시간동안 업데이트된 데이터가 많아 클라이언트 캐시 데이터의 대부분을 유지할 수 없다고 판단될 때 전송된다.

본 논문에서 제안하는 기법은 짧은 접속 단절에 대하여 기존의 방법대로 시간 의존적인 업데이트 정보를 가지는 CIR을 이용하여 캐시 일관성을 유지한다. 그러나 장기간의 접속 단절에 대해서는 클라이언트가 접속 단절 시점과 캐시에 대한 전체적인 정보만을 전송한다. 이 정보를 받은 서버는 클라이언트의 접속 단절 시간 동안의 업데이트를 확인함으로써 클라이언트의 캐시에서 어느 정도의 데이터를 유지할 수 있는지를 예측하여 캐시 유지비용이 적은 경우라면 RIR을, 그렇지 않다면 DM을 전송하는 방법을 사용한다. 여기에서 RIR과 DM은 시간 의존적인 정보가 아닌 서버의 업데이트 정도에 따라 결정되는 IR이다.

3. 클라이언트 작업모델

클라이언트의 작업은 크게 데이터 캐시, 서버로의 요청, 질의 처리의 세 경우로 구분할 수 있다.

3.1 클라이언트의 데이터 캐시

서버는 전체 클라이언트의 요구에 따라 전체 데이터가 아닌 일부의 데이터를 브로드캐스트 하는데 이 때 전송 시점의 timestamp, 데이터 이름, 값, 브로드캐스트 빈도가 함께 전송된다. 브로드캐스트를 받은 클라이언트는 자주 사용하는 데이터를 캐시하는데 이 때 데이터의 이름, 값, 브로드캐스트 빈도, 브로드캐스트 된 timestamp를 함께 캐시한다.

3.2 클라이언트의 서버로의 요청

클라이언트의 서버로의 요청은 크게 두 가지로 구분할 수 있다.

첫째는 데이터 브로드캐스트 요청(Broadcast Request : BR)이다. 서버는 서버의 전체 데이터가 아닌 일부 데이터를 브로드캐스트 하므로 어떤 클라이언트에서 브로드캐스트 되고 있지 않은 데이터가 필요한 경우 클라이언트는 서버에 BR을 전송한다. 따라서 이러한 BR의 경향에 따라 브로드캐스트 스케줄은 계속적으로 변경될 수 있다.

둘째는 캐시 데이터에 대한 일관성 확인 요청(Validation Request : VR)으로, 클라이언트가 일정 시간 이상 접속 단절되어 CIR 만으로 캐시 일관성 확인이 불가능한 경우 서버에 전송하는 것이다.

BR과 VR은 아래와 같은 항목을 가진다.

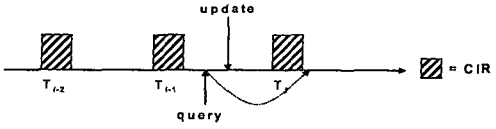
- *BR : <Name>*
- *VR : <Last_TS, Num_CD(n), L_TS(n), G_TS(n)>*
  - Last_TS : 클라이언트가 접속 단절되기 이전에 마지막으로 수신한 CIR의 tiemstamp
  - Num_CD(n) : 클라이언트 캐시에 저장된 데이터 가운데, 브로드캐스트 빈도가 n인 데이터의 수
  - G_TS(n) : 클라이언트 캐시에 저장된 데이터 가운데, 브로드캐스트 빈도가 n이며, 가장 큰 timestamp를 가지는 데이터의 timestamp 값
  - L_TS(n) : 클라이언트 캐시에 저장된 데이터 가운데, 브로드캐스트 빈도가 n이며, 가장 작은 timestamp를 가지는 데이터의 timestamp 값

3.3 클라이언트 내의 질의 처리

클라이언트는 질의를 처리하기 위하여 특정 데이터를 필요로 하며 필요한 데이터가 캐시에 저장되어 있지 않다면 브로드캐스트 되는 데이터를 수신하여 질의 처리를 수행해야 하고, 저장되어 있다면 데이터의 일관성을 확인해야 한다. 그런데 이러한 일관성 확인은 클라이언트의 접속 단절 여부와 접속 단절 동안 수신하지 못한 CIR이 있는지 없는지에 따라 아래와 같이 구분될 수 있다.

• 접속 단절되지 않은 경우

클라이언트 내에서 질의가 발생한 경우 필요한 데이터가 캐시에 저장되어 있지 않다면, 클라이언트는 서버에서 브로드캐스트되는 데이터를 수신함으로써 질의 처리가 가능하다. 또한 필요한 데이터가 캐시에 저장되어 있다면, 질의는 즉시 수행되지 않고 그림 1과 같이 다음 CIR을 수신할 때까지 기다렸다가 CIR 수신 결과 캐시의 데이터가 유효하다고 판단되면 캐시의 데이터의 timestamp를 CIR의 timestamp로 업데이트한 후 해당 데이터를 이용하여 질의를 처리할 수 있다. 만약, CIR 수신 결과 캐시의 데이터가 유효하지 않다면 이러한 데이터는 버리고 다음 브로드캐스트되는 데이터를 수신하여 질의 처리를 수행해야 한다.



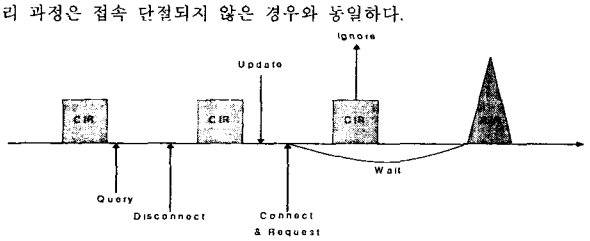
[그림 1] 클라이언트의 질의 처리

• 접속 단절되었으나 수신하지 못한 CIR이 없는 경우

재 연결된 클라이언트는 다음 번 CIR을 수신할 때 까지 아무런 작업도 수행하지 않는다. 재 연결 이후 처음으로 수신한 CIR의 timestamp 값을 Cur_TS라고 할 때, Cur_TS와 Last_TS 값의 차가 L보다 작다면 이는 수신하지 못한 CIR이 없음을 의미하므로 위의 접속 단절되지 않은 경우와 동일한 과정을 통하여 질의 처리가 가능하다.

• 접속 단절되었으며 하나 이상의 CIR을 수신하지 못한 경우

Cur_TS와 Last_TS 값의 차가 L보다 크다면 이는 수신하지 못한 CIR이 하나 이상 존재함을 의미하므로 CIR 만으로 캐시 데이터 일관성을 확인할 수 없다. 이러한 경우에 클라이언트는 VR을 전송한다. VR의 전송 결과 서버에서 RIR 혹은 DM이 전송되며 클라이언트는 RIR을 수신한 경우 업데이트된 데이터를 버리고, DM을 수신한 경우 모든 데이터를 버린다. 이와 같은 캐시의 일관성 유지 이후의 질의 처리 과정은 접속 단절되지 않은 경우와 동일하다.



[그림 2] 클라이언트의 RIR 수신

4. 서버 작업모델

본 논문에서는 클라이언트, 서버 환경에서 동시에 하나의 클라이언트만이 접속 단절되는 경우를 가정한다. 또한 서버의 전체 데이터가 아닌 일부의 데이터가 브로드캐스트 되며, BR에 따라 브로드캐스트 스케줄을 변경하는 환경을 가정한다. 이에 따라 서버에서는 브로드캐스트된 데이터와 업데이트 된 데이터에 대한 리스트를 계속적으로 유지하여야 한다. 또한 VR의 수신 여부에 따라 CIR, RIR, DM의 전송 여부를 결정하여야 한다.

4.1 서버의 리스트 유지

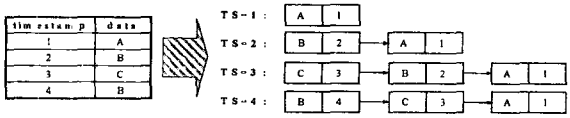
서버는 VR 처리를 위하여 브로드캐스트 리스트(Broadcast List : BL)와 업데이트된 데이터의 리스트(Update List : UL)를 유지한다.

• 브로드캐스트 리스트

각 데이터에 대하여 서버에서 브로드캐스트 할 때 브로드캐스트 tiemstamp를 함께 유지한다. 예를 들어 D1, D2 두 개의 데이터가 있고 D1은 timestamp 1부터 10사이, D2는 5부터 15사이에 브로드캐스트 되었다면 D1 : 1-10 / D2 : 5-15와 같이 브로드캐스트된 데이터에 대해 timestamp를 함께 유지한다. BL에서 한번의 전체 주기에 해당되는 전체 데이터가 업데이트 된 경우에는 리스트의 해당 항목들을 삭제한다.

• 업데이트 리스트

서버는 각 데이터에 대하여 가장 최근에 업데이트 된 timestamp 값을 가지는 리스트를 유지한다. 그림 3은 각 시간에 업데이트된 데이터에 대하여 유지해야 할 리스트를 시간별로 나타낸 것이다.



[그림 3] 서버의 업데이트 리스트(UL) 유지

4.2 전송 IR 결정

서버의 수행 작업은 데이터 브로드캐스트, CIR 전송, RIR 전송, DM 전송은 네 가지로 구분되며 각 작업은 아래와 같이 구분되어 수행된다.

• 클라이언트에게 VR을 받지 않았을 때

서버는 사용자의 BR 경향에 따라 브로드캐스트 스케줄을 변경하며 L 주기마다 CIR을 전송한다.

• 클라이언트에게 VR을 받았을 때

서버는 아래와 같은 과정을 거쳐 RIR을 전송할 것인지, DM을 전송할 것인지를 판단한다.

- *Step 1*
  L_TS(n)보다 크거나 같고 G_TS(n)보다 작거나 같은 timestamp 값을 가지는 데이터를 BL에서 검색한다. 이와 같은 과정을 거쳐 검색된 데이터의 수를 Num_SD(n)이라 한다.
- *Step 2*
  Step 1에서 검색된 데이터 가운데 Last_TS 보다 큰 timestamp 값을 가지는 데이터를 UL에서 검색한다. 이와 같은 과정을 거쳐 검색된 데이터의 수를 Num_SU(n)이라 한다.
- *Step 3*
  Num_SD(n):Num_SU(n)=Num_CD(n):Num_CU(n)의 비례식을 이용하여 Num_CU(n)을 구한다.
- *Step 4*
  서버에서 한 번의 전체 주기 동안 브로드캐스트하는 데이터들의 크기(Size_Bcast)와 step 2에서 검색된 데이터 전체를 이용하여 구성한 RIR의 크기(Size_RIR)에 따라 β를 계산한다.

$$\beta = 1 + \frac{Size\_RIR}{Size\_Bcast + Size\_RIR}$$

- *Step 5*
  Cost_RIR < Cost_DM : RIR 전송
  Cost_RIR ≥ Cost_DM : DM 전송

$$Cost\_cache = \sum (Num\_CU(i) \times \sum \frac{j \times Broadcast\_SD(j)}{\quad})$$

$$Cost\_send = \sum Num\_SU(k)$$

$$Cost\_RIR = \beta(Cost\_cache + Cost\_send)$$

$$Cost\_DM = \sum_{i}(Num\_CD(i) \times \frac{\sum_{j} j \times Bcast\_SD(j)}{2i})$$

α : 하나의 데이터 브로드캐스트 패킷 안에 들어갈 수 있는 RIR 데이터의 개수

Bcast_SD(n) : 한 번의 전체 주기 동안 n의 빈도로 브로드캐스트 되는 데이터의 개수

Step 1의 *L_TS(n), G_TS(n)*는 클라이언트가 캐시하고 있는 데이터가 언제 브로드캐스트 된 데이터인지에 대한 정보를 나타낸다. 서버는 매 브로드캐스트 스케줄마다 일부의 데이터를 브로드캐스트 하므로 클라이언트가 캐시하고 있는 데이터의 범위를 확인함으로써 업데이트 여부를 확인해야 하는 데이터의 양을 줄일 수 있다.

Step 2는 클라이언트가 언제까지의 업데이트를 반영하고 있는지 확인하는 단계로, 서버는 *Last_TS* 이후 업데이트에 대해서만 확인한다.

Step 3는 *Num_SD(n)* 중에서, 얼만큼의 데이터가 업데이트 되었는지의 비율에 따라 실제 클라이언트가 캐시하고 있는 데이터 가운데 어느 정도의 데이터가 업데이트 되었는지를 예측하는 단계이다.

Step 4는 전체 대역폭에서 RIR이 차지하는 비중을 계산하기 위한 단계로서 예를 들어 RIR이 차지하는 비율이 10%라면 β=1.1이 된다.

Step 5에서 *Cost_cache*는 클라이언트 캐시에서 업데이트된 데이터를 버린 후 브로드캐스트 채널을 통하여 다시 캐시하는데 필요한 시간이며 *Cost_send*는 RIR을 전송하는데 필요한 시간이다. 결국 *Cost_cache+Cost_send*는 RIR을 전송할 경우 클라이언트 캐시를 다시 채우는데 필요한 전체 시간이다. *Cost_DM*은 모든 데이터를 버린 후 캐시를 다시 채우는데 필요한 시간이다. 따라서 *Cost_cache+Cost_send* < *Cost_DM*인 경우 RIR을 전송하면 시간적인 이득이 있다고 볼 수 있다. 그러나 아주 작은 시간적 이득을 위하여 큰 대역폭을 사용하는 것은 옳지 못하므로 *Cost_cache+Cost_send*에 β를 곱하여, RIR을 전송함으로써 얻을 수 있는 시간적인 이득이 전체 대역폭에서 RIR이 차지하는 비중 이상일 경우에 RIR을 전송함을 의미한다.

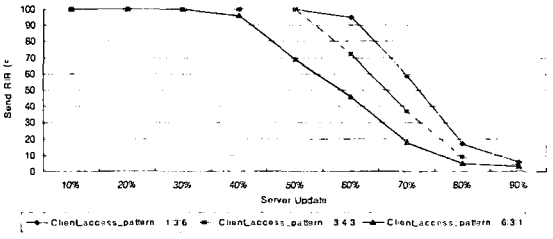
5. 성능 평가

본 논문에서는 제안된 방법은 서버의 업데이트가 많고, 온도가 높은 데이터를 캐시한 클라이언트일수록 RIR 전송 비율이 낮아져야 한다. 따라서 이러한 상황에서 RIR 전송 비율이 효율적으로 결정되는지 판단하기 위하여 표 1과 같은 환경에서의 성능 평가를 실시하였다.

[표 1] 성능 평가를 위한 변수

항 목	변수 값 설정	내 용
Total_SD	1100개	서버의 전체 데이터 수
Bcast	550개	한 번의 스케줄에의하여 브로드캐스트되는 전체 데이터 수
Change_access_pattern	10%	BR에 의한 브로드캐스트 스케줄의 변화 정도
Update	10%, 20%, …, 80%, 90%	매 브로드캐스트 주기마다의 서버 업데이트 정도
Bcast_freq	4:2:1	브로드캐스트 빈도
α	2	하나의 데이터 브로드캐스트 패킷 안에 들어갈 수 있는 RIR 데이터의 개수
Size_cache	100	클라이언트 캐시 크기
Client_access_pattern	6:3:1, 3:4:3, 1:3:6	데이터의 브로드캐스트 빈도에 따른 클라이언트의 데이터 캐시 비율
Term_disconn.	10	클라이언트의 접속 단절 시간

본 논문에서는 서버의 일부 데이터가 브로드캐스트되는 환경을 가정하므로 이에 따라 *Total_SD, Bcast, Change_access_pattern*을 설정하였다. 오랜 시간 브로드캐스트 되는 환경을 가정할 때 서버의 전체 데이터가 한 번 이상씩 브로드캐스트 된다고 생각할 수 있으므로 이 값은 RIR의 전송 여부에 큰 영향을 미치지 않는다. *Bcast_freq*는 4:2:1로 가장 온도가 높은 데이터가 4번 브로드캐스트 될 동안 가장 온도가 낮은 데이터는 한 번 브로드캐스트됨을 의미한다. α는 일반적으로 값이 클수록 RIR 전송 비율이 높아짐을 예측할 수 있다. 클라이언트의 캐시 크기는 임의로 100을 설정하였으며, *Term_disconn*은 그 시간이 길어질 수록 RIR 전송 비율이 달라질 것이나 *Term_disconn*이 길어짐은 *Update*가 증가함과 같은 의미로 해석할 수 있으므로 여기에서는 *Update* 만을 변경시켜 성능 평가를 수행하였다. *Client_access_pattern*은 클라이언트가 어떤 데이터를 주로 캐시하는지를 나타내는 값으로 6:3:1은 브로드캐스트 빈도 4인 데이터를 빈도 2인 데이터에 비하여 2배, 빈도 1인 데이터에 비하여 6배 많이 캐시하고 있음을 의미한다. 본 논문에서 제안된 기법은 클라이언트가 어느 브로드캐스트 빈도를 가지는 데이터를 주로 캐시하는지에 따라 RIR 전송 여부에 큰 영향을 미친다. 따라서 온도가 높은 데이터를 주로 캐시하는 클라이언트와 낮은 데이터를 주로 캐시하는 클라이언트, 전체적으로 고르게 캐시하는 클라이언트에 따라 RIR 전송 비율을 측정하기 위하여 6:3:1, 1:3:6, 3:4:3의 캐시 성향에 대하여 그 변화를 살펴보았다. 표1과 같은 경우 그림 4의 결과를 나타내었다.



[그림 4] RIR 전송 비율

위의 결과에서도 볼 수 있듯이 본 논문에서 제안된 RIR 전송에 따른 클라이언트 캐시 일관성 유지 기법은 같은 데이터를 캐시하고 있는 클라이언트의 경우, 서버에서 업데이트가 많을수록 RIR 전송 비율이 낮아진다. 또한, 같은 업데이트 정도인 경우 브로드캐스트 빈도가 높은 데이터를 캐시하고 있는 클라이언트가 그렇지 않은 클라이언트보다 RIR을 전송받는 비율이 낮아진다. 이는 서버에서 업데이트가 많이 발생할수록 확률적으로 클라이언트 캐시에서 유지할 수 있는 데이터가 적어지며, 브로드캐스트 빈도가 높은 데이터를 캐시하고 있는 클라이언트일수록 캐시한 데이터를 버린 이후 다시 캐시하기 위해 걸리는 시간이 적게 걸린다는 면에서 적절한 결과임을 알 수 있다.

6. 결론 및 향후 연구

본 논문에서는 stateless 서버 환경에서 빈번한 접속 단절로 인하여 IR을 수신하지 못한 클라이언트가 서버의 업데이트 정도에 따라 효율적으로 캐시 일관성을 유지할 수 있는 기법을 제안하였다. 본 논문에서 제안된 기법은 클라이언트 캐시 일관성 유지를 위하여 접속단절 동안의 서버 업데이트 정도와 클라이언트의 데이터 캐시 성향을 함께 고려한다. 이를 위하여 서버에서는 업데이트와 브로드캐스트 리스트를 유지해야 하는 부담이 생기지만 장시간의 클라이언트 접속 단절에 대해서도 업데이트 정도에 따라 동적으로 클라이언트 캐시를 유지할 수 있으며, RIR 전송 시 업데이트된 모든 데이터가 아니라 클라이언트가 캐시하고 있을 가능성이 있는 데이터만을 전송함으로써 모든 업데이트 정보를 전송하는 경우에 비하여 IR의 크기를 줄일 수 있다는 장점을 가지고 있다. 이와 같은 특징은 성능 평가를 통해 확인할 수 있는데, 성능 평가 결과 서버의 업데이트가 많고, 온도가 높은 데이터를 캐시한 클라이언트일수록 RIR 전송 비율이 낮아짐을 확인할 수 있다.

현재까지의 연구는 특정 클라이언트의 캐시에 저장되어 있는 데이터의 특성에 기반하여 일관성 유지를 수행하므로 동시에 하나 이상의 클라이언트의 접속 단절 환경에서는 대역폭이나 서버의 작업 수행에 많은 부담이 생긴다. 따라서 향후 연구에서는 동시에 다수의 클라이언트가 접속 단절되는 경우에 대하여 클라이언트의 일관성 유지를 수행할 수 있는 방법에 대한 연구가 필요하다.

7. 참고 문헌

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

[2] D.Barbara and Tomasz Imielinski, "Sleepers and workaholics: caching strategies in mobile environments" ACM SIGMOD Record, Proceedings of the 1994

[3] Qinglong Hu and Dik Lun Lee, "Adaptive cache invalidation methods in mobile environments", High Performance Distributed Computing, 1997. Proceedings. The Sixth IEEE International Symposium on , 1997. pp.264-273