DOI: 10.3745/KIPSTA.2009.16-A.3.181

트랜스코딩 프록시에서 트랜스코딩 부하를 줄이기 위한 낮은 품질 서비스 정책

박 유 현†

요 약

트랜스코딩은 서비스 품질(QoS)에 따른 VoD 서비스를 제공하기 위해 필요한 기본 기술 중 하나이지만, CPU 자원이 많이 필요하다는 단점을 가진다. 트랜스코딩 프록시는 멀티미디어 데이터를 트랜스코딩 할 뿐만 아니라 미래의 사용에 대비하기 위해 캐싱도 할 수 있는 프록시 서버이다. 본 논문에서는 트랜스코딩 프록시에서 서비스 품질을 낮춤으로써 트랜스코딩 부하를 줄이는 서비스 정책을 제안한다. 서비스 품질과 시스템 부하는 tradeoff 관계에 있기 때문에, 사용자가 요청한 품질보다 낮은 서비스 품질을 제공함으로써, 트랜스코딩 프록시는 보다 많은 사용자에게 서비스를 제공할 수 있다.

키워드 : 낮은 품질 서비스, 트랜스코딩, 멀티미디어 캐싱, 세그먼트 기반 캐싱

A Degraded Quality Service Policy for reducing the transcoding loads in a Transcoding Proxy

Park Yoohyun†

ABSTRACT

Transcoding is one of core techniques that implement VoD services according to QoS. But it consumes a lot of CPU resource. A transcoding proxy transcodes multimedia objects to meet requirements of various mobile devices and caches them to reuse later. In this paper, we propose a service policy that reduces the load of transcoding multimedia objects by degrading QoS in a transcoding proxy. Due to the tradeoff between QoS and the load of a proxy system, a transcoding proxy provides lower QoS than a client's requirement so that it can accomodate more clients.

Keywords : Degraded Quality Service, Transcoding, Multimedia Caching, Segment-Based Caching

1. 서 론

최근의 사용자들은 유선망뿐만 아니라 무선망을 통해서도 서비스를 제공 받으며, 특히 무선망을 사용하는 단말기(device)들은 노트북과 같은 비교적 고성능에서부터 휴대전화와 같은 저성능 단말기까지 그 종류가 다양하기 때문에 성능, 대역폭, 화면 크기 등의 시스템 자원을 고려한 적응형 서비스(adaptive service)가 필수적이다[1]. 적응형 서비스를 제공하기 위한 방법은 정적 적응방법, SVC(Scalable Video Coding) 방법, 그리고 실시간 트랜스코딩 방법이 있는데, 트랜스코딩 방법은 사용자의 상태에 가장 적합한 컨텐츠 품질을 생성할 수 있는 방법이다. 이러한 컨텐츠 적응(content adaptation) 기법과 캐시를 결합하면 네트워크 자원 및 서버의 부하를 줄이는 한편, 다양한 단말에 대한 컨텐츠 적응 기능을 제공할 수 있다. 적응형 서비스를 제공할 수 있는 프록시의 대표적인 예가 트랜스코딩 프록시(transcoding proxy)인데, 트랜스코딩 프록시는 서버와 클라이언트 사이에서 캐싱 기능과 트랜스코딩을 수행한다.

트랜스코딩은 동영상의 경우 다른 규격(포맷)으로 변환하거나 전송률, 해상도, 화면크기 등을 조절하는 것을 의미하며, 이러한 변환을 위해서는 낮은 수준의 계산을 반복적으로 수행해야 하기 때문에 CPU 자원을 과도하게 사용하는 문제점을 가진다[1, 2]. 이러한 문제점을 해결하기 위해서는 저장장치를 매우 크게 확장시켜 트랜스코딩된 결과를 계속 캐싱하는 방법과 다중 클러스터로 트랜스코딩 서버들을 묶어서 CPU 자원과 저장장치를 확장시키는 방법[3, 4], 그리고 트랜스코딩 작업의 중단 단계 데이터(metadata)를 캐싱하여 적은 CPU 자원으로 트랜스코딩을 수행하는 방법[2]들

† 정 회 원 : 동의대학교 컴퓨터소프트웨어공학과 전임강사
논문접수 : 2008년 11월 11일
수 정 일 : 1차 2009년 1월 6일, 2차 2009년 2월 13일
심사완료 : 2009년 2월 20일

이 연구되고 있다.

트랜스코딩은 시스템 자원이 많이 필요한 연산이기 때문에, 여러 개의 트랜스코딩 작업(job)을 동시에 수행하게 되면 CPU 자원의 부족으로 수행되지 못하는 트랜스코딩 작업이 존재할 수 있다. 이때, 해당 트랜스코딩 작업을 통하여 서비스를 제공 받는 사용자는 CPU 자원을 할당 받을 때까지 기다려야 하고, 사용자 측에서는 지터가 발생하며, 만일 대기 시간이 길어진다면 사용자 이탈의 가능성도 높아진다. 이러한 문제점을 해결하기 위해서 프록시 관리자는 사용자가 요청한 컨텐츠의 해당 버전이 캐싱되지 않아서 트랜스코딩으로 해당 버전을 생성해야 하는 경우에, 요청한 버전을 위한 트랜스코딩을 수행하지 않고, 사용자가 원하는 버전보다 낮은 품질의 버전이 이미 캐싱되어 있다면 이를 서비스하는 정책을 선택할 수 있다. 이러한 방법은 사용자의 품질 만족도를 일부 떨어뜨릴 수 있으나, 사용자 지연시간을 줄이고, 시스템의 부하를 줄이는 장점을 가진다.

본 논문에서는 이와 같이 요청된 품질의 버전이 캐싱되어 있지 않지만, 이보다 낮은 품질의 버전을 캐싱하고 있다면 이 버전으로 서비스를 제공하는 방법을 낮은 품질 서비스(DQS:Degraded Quality Service)로 정의하고, 낮은 품질 서비스의 적용으로 얻을 수 있는 이점을 실험을 통하여 제시한다. 실험에 의하면 낮은 품질 서비스 방식을 채택하면 채택하지 않을 때에 비하여 DSR은 최대 11%, BHR는 최대 24% 향상되고, 트랜스코딩 되는 데이터는 최대 7% 줄이는 장점을 가진다. 따라서, 낮은 품질 서비스 정책을 채택하면 보다 적은 시스템 자원을 활용하여 보다 많은 사용자에게 서비스를 제공할 수 있기 때문에 사용자가 요청한 품질 기준을 꼭 만족하지 않아도 되는 경우에는 트랜스코딩 프록시를 통하여 더 많은 사용자에게 스트리밍 서비스를 제공할 수 있다.

2. 관련 연구

2.1 트랜스코딩 프록시

초기의 트랜스코딩 프록시에 관한 연구들은 주로 웹 문서만을 대상으로 하였으나 최근 연구들은 점차 오디오, 비디오 데이터를 대상으로 점차 확대되고 있는데, 웹 문서에 비해 오디오, 비디오 데이터는 크기가 매우 크고 트랜스코딩을 수행할 때 더 많은 시스템 자원을 사용하는 특징을 가진다.

트랜스코딩과 프록시를 접목시킨 연구로는 FVO/TVO[5], TeC[6], AE[7], VPF[8], PT-2[9]등이 있다.

FVO(Full Version Only)는 트랜스코딩 원본 버전만을 캐싱하는 방법이고, TVO(Transcoded Version Only)는 트랜스코딩을 수행한 결과 버전만을 캐싱하는 방법이다. 따라서, 원본 버전을 캐싱하는 FVO는 네트워크 자원을 소비하여 원본 서버로부터 데이터를 가져오는 빈도가 적어지나 낮은 품질 버전을 서비스 하기 위해서는 항상 트랜스코딩을 수행해야 하기 때문에 CPU 자원은 많이 소모하게 되고, TVO는 반대로 네트워크 자원을 비교적 많이 사용하게 된다.

또한 TeC(Transcoding enable Caching)는 트랜스코딩 프록시의 교체함수로, [6]에서는 LRU 방법을 사용하여 특정 시간에는 하나의 버전만을 캐싱하는 TeC11, TeC12와 동시에 여러 개의 버전을 캐싱할 수 있는 TeC2를 제안하였다.

[9]은 TeC와 마찬가지로, 트랜스코딩 프록시의 교체정책(replacement policy)으로 LRU를 사용하지만, 객체를 반씩 나누어 관리하는 부분 캐싱 방법을 제안하였다. 즉, 희생 데이터로 선택이 되면 먼저 전체 데이터의 뒤에 있는 반을 버리고, 다음 번 희생 데이터로 선택이 되었을 때 앞의 반을 버리는 정책이다.

[7]에서는 웹 데이터를 위한 프록시에서 같은 컨텐츠의 다중 버전 사이의 관계를 트랜스코딩 그래프로 표현하고 트랜스코딩 그래프를 이용하여 이득값(profit)를 계산하여 이득값을 기반으로 교체정책을 수행하였다. 이 방법은 기존의 LRU를 사용하는 방법에 비해 우수한 성능을 보인다.

[8]은 [7]의 방법을 비디오 데이터에 적용하기 위해서 평균 참조 길이와 트랜스코딩 길이를 이용하여 이득값 계산 수식을 개선하여 VPF(Video Profit Function)을 교체함수로 사용하였다.

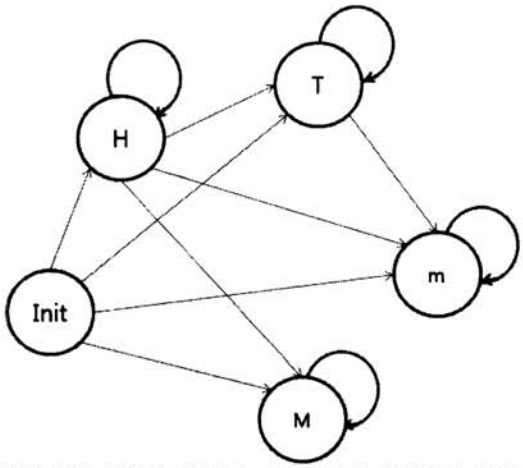
또한, [10]에서는 부분 캐싱을 하는 트랜스코딩 프록시에서의 이벤트들을 정의하고, 부분 캐싱을 효과적으로 제공하기 위해 컨텐츠를 세그먼트(segment) 단위로 관리하는 방법을 제안하였다.

2.2 트랜스코딩 이벤트

일반적인 프록시 시스템에서는 히트(hit)와 미스(miss)라는 두 가지 이벤트가 사용되는데, 두 개의 이벤트들은 프록시 시스템이 어떤 동작을 수행해야 하는지를 결정한다. 즉, 프록시 서버는 히트 이벤트가 발생하면 즉시 사용자에게 요청한 데이터를 전송하고, 미스 이벤트가 발생한 경우에는 먼저 서버에게 사용자 요청을 전달하여 데이터를 전송 받아(fetch) 프록시 서버에 저장하고 이를 사용자에게 다시 전달한다.

트랜스코딩 프록시는 이러한 경우 외에 사용자가 특정 버전(V_r)을 요청했을 때, 프록시에는 해당 버전은 저장하고 있지 않으나, 이보다 높은 품질의 버전(V_h, $Q(V_h) > Q(V_r)$, $Q(V_r)$는 버전 V_r의 품질)을 저장하고 있어서 높은 품질의 버전을 통하여 요청한 버전을 만들 수 있는 경우(트랜스코딩이 가능한 경우)에 기존의 프록시와는 다른 동작을 하게 된다. 이 경우에 캐시 시스템(cache system)은 트랜스코더(transcoder)에게 트랜스코딩 작업을 요청하고 이의 결과를 사용자에게 전송하여 서비스를 제공하게 된다. 이러한 경우를 기존의 프록시에서 사용하는 히트, 미스와 구별되도록 트랜스코딩 히트(transcoding hit)라고 정의한다 [6, 8].

만일 트랜스코딩 프록시가 기존의 웹 프록시와 같이 객체 단위로 캐싱하게 되면 히트, 미스, 트랜스코딩 히트의 세 가지 이벤트로 모든 경우를 표현할 수도 있다. 하지만, 트랜스코딩 프록시에서 주로 다루고자 하는 미디어는 비디오, 오



(그림 1) 연속 캐싱을 사용하는 트랜스코딩 프록시의 이벤트 전이 그래프

디오와 같은 대용량 미디어이기 때문에, 객체 단위로 캐싱할 경우 프록시의 저장공간의 급격한 소모로 인해 캐싱 효율이 떨어지게 된다. 따라서 트랜스코딩 프록시의 성능을 높이기 위해서는 부분 캐싱을 적용하는 것이 유리한데, 이 경우에 기존의 세 가지 이벤트로는 표현이 불가능한 경우가 발생한다.

[9]에서는 부분 캐싱에서 발생하는 이벤트들을 FF(Full Fetch), FH(Full Hit), FT(Full Transcoding), FFT(Full Fetch Transcoding), PHF(Partial Hit and Fetch), PTFT (Partial Transcoding and Fetch Transcoding)로 세분화하여 정의하였다. 하지만, 이 연구에서도 몇 가지 경우에 대해서 표현할 수 없는 경우가 발생하였다.

[10]에서는 부분 캐싱을 지원하는 트랜스코딩 프록시에서 (그림 1)의 H, T, M, m 이벤트로 구성된 이벤트 전이 그래프를 이용하여 H, T, M, m, HT, HM, Hm, HTm, Tm의 9개 이벤트 시퀀스를 정의하였다.

3. 낮은 품질 서비스 정책

본 장에서는 트랜스코딩 프록시에서 사용자의 요청에 대한 서비스를 제공하는 정책에 대하여 설명한다. 먼저, 시스템 구조와 서비스 모델을 살펴본 후, 낮은 품질 서비스 (DQS : Degraded Quality Serivce) 정책의 구체적인 방법을 설명한다.

사용자의 스트리밍 요청에 대해서 서비스를 제공하는 방법은 크게 사용자 중심과 시스템 중심 정책으로 나눌 수 있다. 사용자 중심 서비스 정책은 사용자의 요청에 충실한 서비스를 제공하는 것이다. 즉, 사용자가 고품질 버전(version)을 요청하면, 고품질 버전을 제공하고, 저품질 버전을 요청하면 저품질 버전을 제공하면 된다. 하지만, 품질 변환을 위해서는 트랜스코딩이 필요하고 트랜스코딩은 고비용의 연산이 필요하기 때문에, 제한된 시스템 자원으로는 사용자 중심 서비스를 항상 제공할 수는 없다. 이에 반하여, 시스템 중심 정책은 사용자 요청에 대하여 시스템의 자원을 적게 사용하여 서비스를 제공하는 방식이다. 이 방법은 사용자 만족도를 떨어뜨릴 수 있으나, 자원 소모를 적게 하여 보다 많은 사용자에게 서비스를 제공할 수 있게 된다. 따라서, 서비스 비용을 충실히 제공하는 사용자를 위해서는 사용자 중심의 서비스를 제공해야 하지만, 무료 서비스 또는 저렴한 사용료를 지불하는 사용자를 위해서는 시스템 중심의 서비스 정책을 채택하는 것이 바람직하다.

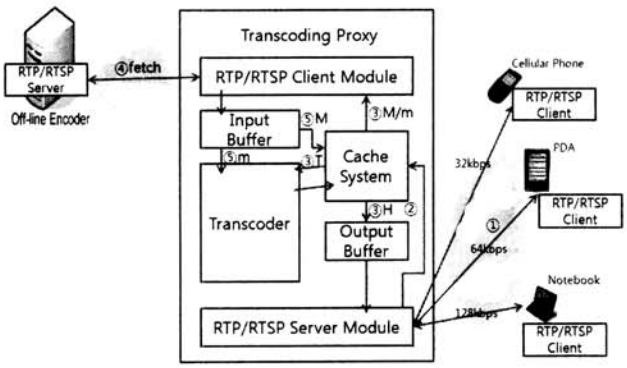
본 논문은 이러한 시스템 중심의 서비스 정책에 관한 구체적인 내용에 관해 설명한다. 시스템 중심의 서비스 정책에 관하여 [1]에서 secondary hit란 이름으로 소개한 적이 있으나, 이 연구에서는 간단히 언급하는 정도로 설명하여, 부분 캐싱을 적용하지 않았고, 낮은 품질 버전이 여러 개 있을 경우의 선택기준을 설명하지 않았으며, 실제 어떤 성능을 갖는지에 대한 구체적인 실험을 수행하지 않았다. 따라서, 본 논문에서는 [1]에서 추상적으로 설명했던 내용에 대해 서비스 모델과 구체적인 방법을 설명하고, 그 실험결과를 제시하고자 한다.

3.1 시스템 구조

트랜스코딩 프록시 사용하는 컨텐츠는 일반 프록시에서 사용하는 컨텐츠와는 다른 특징을 가진다. 즉, 하나의 컨텐츠에 대해서도 다양한 전송률(bitrate)를 가지는 여러 버전(version)이 존재할 수 있는데, 높은 전송률을 가지는 버전을 이용하여 낮은 전송률의 버전을 트랜스코딩 작업으로 만들 수 있다.

일반적인 트랜스코딩 프록시의 구조는 (그림 2)와 같으며, 이 구조는 일반적인 스트리밍 프록시 구조에서 트랜스코딩 모듈이 추가된 모습이다. 트랜스코딩 프록시는 H, T, M, m의 기본 이벤트로 동작하며, RTP/RTSP 서버 모듈이 사용자 단말로부터 서비스 요청을 받고 캐시 시스템에서 요청받은 데이터의 저장 여부를 판단하여 저장 되어 있지 않다면 프록시의 RTP/RTSP 클라이언트 모듈을 통해 원본 서버에게 데이터를 요청하고(M, m 이벤트), 전달 받은 데이터를 바로 사용자 단말에게 전달하거나(M 이벤트), 추가적으로 트랜스코딩 작업이 필요한 경우(m 이벤트)에는 트랜스코딩을 수행한 결과를 사용자 단말에게 전달한다. 또한, 이미 저장되어 있는 데이터인 경우에는 원본 서버의 개입 없이 프록시가 직접 사용자 단말에게 서비스를 제공한다(H 이벤트). 만일, 사용자가 요청한 데이터는 캐시되어 있지 않으나, 그보다 높은 품질의 데이터를 캐시하고 있는 경우에는 프록시가 트랜스코딩을 수행하고 그 결과 데이터로 서비스를 제공한다 (T 이벤트). 따라서, M 이벤트에 필요한 동작과 T 이벤트에 필요한 동작을 모두 수행하는 것이 m 이벤트이다[10].

만일, 캐싱해야 할 데이터가 이미지와 같이 객체 단위로 캐싱하는 데이터들은 4개의 기본 이벤트로 동작을 설명할 수 있지만, 오디오, 비디오와 같이 부분 캐싱을 하는 방법에서는 4개의 기본 이벤트를 시간 순으로 조합하여 총 9개의 이벤트 시퀀스를 이용해야 한다. 즉, 앞부분은 캐싱되어 있



(그림 2) 트랜스코딩 프록시의 구조

으나, 뒷부분은 없는 경우(HM, Hm), 앞부분은 캐싱되어 있으나 뒷부분은 트랜스코딩 해야 하는 경우(HT), 앞부분은 캐싱되어 있고, 중간부분은 트랜스코딩 해야 하지만, 뒷부분은 없는 경우(HTm), 앞부분은 트랜스코딩 해야 하고, 뒷부분은 없는 경우(Tm) 등의 이벤트 시퀀스가 존재하며, 각각의 이벤트에 대한 후속작업을 수행하게 된다[10].

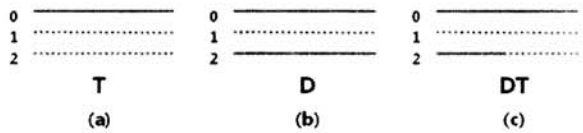
3.2 서비스 모델

DQS 정책은 사용자가 요청한 품질을 만족하는 버전이 캐싱되어 있지 않으나, 그보다 낮은 품질 버전이 캐싱되어 있어 이를 이용하여 서비스를 제공하는 정책이다.

기존 [10]에서 정의된 이벤트 시퀀스 중에서 트랜스코딩 연산이 포함된 이벤트 시퀀스는 *T*, H*T*, H*Tm*, *Tm*, *m*, H*m*이다. 이러한 이벤트가 발생한 경우에 시스템 부하가 임계치를 넘고, 사용자가 요청한 버전보다 낮은 품질의 버전이 캐싱되어 있을 때 D 이벤트를 이용하여 다른 이벤트 시퀀스로 대체된다. 기존의 이벤트 시퀀스가 D 이벤트를 이용한 이벤트 시퀀스로 대체 되면 트랜스코딩을 수행해야 하는 범위를 줄이거나 아예 수행하지 않게 함으로써 시스템 자원을 줄이고, 이에 따라 더 많은 사용자에게 서비스를 제공할 수 있게 된다.

D 이벤트를 이용하여 대체되는 이벤트들은 다음과 같다. 아래 그림들에서 사용자들은 버전 1을 요청한 상태이고, 실선은 캐싱되어 있는 영역을, 점선은 캐싱되지 않은 영역을 나타낸다. 또한 가로축은 시간을 나타내고, (a)의 경우는 기존의 대표적인 이벤트 시퀀스 상황으로 사용자가 요청한 버전 1보다 낮은 버전인 버전 2를 캐싱하고 있지 않아, D 이벤트를 채택하고자 하더라도 D 이벤트를 이용할 수 없는 상황을 나타낸다. 기존의 이벤트들은 사용자가 요청한 버전과 이를 트랜스코딩 하여 생성할 수 있는 상위 버전의 캐싱 여부에 따라 판별 되었지만, D 이벤트를 채택하게 되면, 하위 버전의 캐싱 여부를 살펴야 한다.

먼저, 기존의 T 이벤트는 다음과 같은 경우에 발생한다. 즉, (a)과 같이, 사용자가 요청한 버전 1은 캐싱 되어 있지 않으나, 버전 0을 이미 캐싱하고 있어, 버전 0을 트랜스코딩 하여 사용자에게 버전 1을 서비스 할 수 있는 경우에 T이벤트가 발생한다. 이 중에서도 (a)와 같이 버전 2를 캐싱하

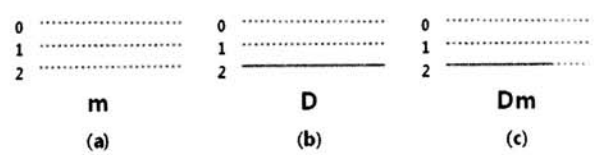


(그림 3) T 이벤트의 대체 이벤트(D, DT)

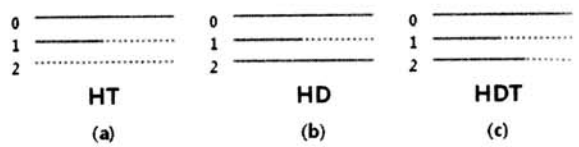
고 있지 않은 경우에는 품질을 낮추어 서비스 할 수 없지만, (b)와 (c)와 같이 버전 2를 일부 또는 전부를 캐싱하고 있는 경우에는 품질을 낮추어 서비스 할 수 있다. (b)와 (c) 각각의 경우에 기존의 T 이벤트로 처리해야 할 사항이 D 이벤트와 DT 이벤트로 대체되어 처리된다. (b)와 같이 T 이벤트가 D 이벤트로 대체되면 트랜스코딩이 전혀 필요 없게 되고, (c)와 같이 T 이벤트가 DT 이벤트로 대체되면 트랜스코딩 해야 할 범위가 줄어들게 되어 결과적으로 트랜스코딩 부하를 줄일 수 있다.

m 이벤트는 (그림 4)와 같이, 버전 0, 1 모두 전혀 캐싱되어 있지 않아 원본 서버로부터 버전 0을 fetch 한 후, 이를 다시 트랜스코딩 하여 사용자에게 전달하는 이벤트이다. 이 경우, 만일 (b)와 같이 전체 영역을 캐싱한 낮은 품질 버전 2가 존재하면 m 이벤트는 D 이벤트로 대체되고, (c) 와 같이 버전 2가 일부 영역만을 캐싱하고 있다면, 캐싱한 부분에 대해서는 D 이벤트로, 버전 2의 나머지 부분에 대해서는 m 이벤트로 처리한다.

HT 이벤트는 (그림 5)와 같이, 버전 1의 일부가 캐싱되어 있으며, 상위 버전인 버전 0는 전체가 캐싱되어 있어, 버전 1의 앞부분은 캐싱되어 있는 부분으로 서비스하고, 뒷부분은 버전 0를 트랜스코딩 하여 서비스하게 된다. 이때, (b)와 같이 버전 2의 전체가 캐싱되어 있다면, 버전 1의 뒷부분을 트랜스코딩 하지 않고 품질을 낮춰 서비스 함으로써 트랜스코딩 연산이 필요 없게 된다. 만일, (c)와 같이 버전 2가 버전 1보다는 더 많이 캐싱하고 있지만, 전체 영역을 캐싱하지 않은 경우에는 버전 1이 캐싱된 부분은 직접 서비스하고(H), 버전 1보다 더 많이 캐싱된 부분에 대해서는 품질을 낮추며(D), 버전2의 나머지 부분에 대해서는 버전 0으로부터 트랜스코딩(T) 하여 트랜스코딩 범위를 줄일 수 있다.



(그림 4) m 이벤트의 대체 이벤트(D, Dm)



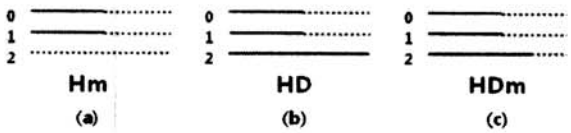
(그림 5) HT 이벤트의 대체 이벤트(HD, HDT)

Hm 이벤트는 (그림 6)과 같이, 버전 1이 일부 캐싱되어 있으며 상위 버전 0는 버전 1과 같거나 작은 영역을 캐싱하고 있을 때 발생한다. 만일, (b)와 같이 버전 2가 전체 영역을 캐싱하고 있다면, 버전 1이 캐싱되어 있지 않은 뒷부분은 버전 2로 품질을 낮춰서 서비스한다. (c) 와 같이 버전 2가 버전 1보다는 많은 영역을 캐싱하고 있지만, 전체 영역이 아닐 경우 버전 2의 캐싱되어 있지 않은 부분에 대해서는 m 이벤트로 처리한다.

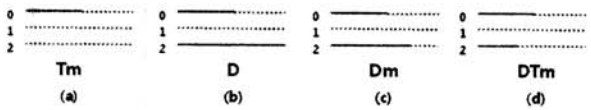
Tm 이벤트는 (그림 7)과 같이 버전 1은 전혀 캐싱되어 있지 않고, 버전 1이 일부 캐싱되어 있을 때 발생한다. 이때, (b)와 같이 버전 2가 전체 영역을 캐싱하고 있다면 D 이벤트로 처리되고, 버전 0보다 큰 영역을 캐싱하고 있다면 Dm 이벤트로, 버전 0보다 적은 영역을 캐싱하고 있다면 DTm 이벤트로 처리된다.

HTm 이벤트 (그림 8)과 같이 버전 1이 일부 캐싱되어 있고, 버전 0는 이보다 더 많은 영역을 캐싱하고 있을 때 발생한다. 이때, (b)와 같이 버전 2가 전체 영역을 캐싱하고 있다면 HD 이벤트로, (c) 와 같이 버전 2가 버전 0보다 큰 영역을 캐싱하고 있다면 HDm 이벤트로, (d)와 같이 버전 2가 버전 0보다는 적은 영역을, 버전 1보다는 큰 영역을 캐싱하고 있다면 HDTm 이벤트로 처리된다.

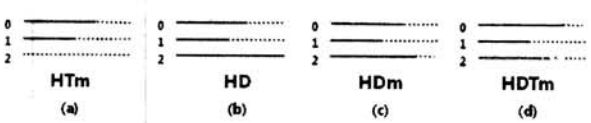
트랜스코딩 연산이 포함된 이벤트 시퀀스 *T*, H*T*, H*Tm*, *Tm*, *m*, H*m*에 대하여 대체 이벤트를 설명한 (그림 3)-(그림 8)의 이벤트 시퀀스들을 효과적으로 표현하기 위해서, 본 논문에서는 [10]에서 제안한 이벤트 전이 그래프를 확장하여 (그림 9)의 낮은 품질 서비스를 지원하는 이벤트 전이 그래프를 제안한다. 제안하는 그래프에서 총 6개의 노드(init, H, T, M, m, D)가 존재하기 때문에 총 화살표는 제안하는 이벤트 전이 그래프에 의하면, D 이벤트로 대체되지 않는 기존의 이벤트 시퀀스 9개(H, T, M, m, HT, HM, Hm, HTm, Tm)와 부하가 큰 트랜스코딩 연산을 낮은 품질의 버전으로 서비스 하는 D 이벤트를 포함한 D, DT, Dm, DTm, HD, HDT, HDm, HDTm의 8개의 이벤트 시퀀스가 존재하여 총



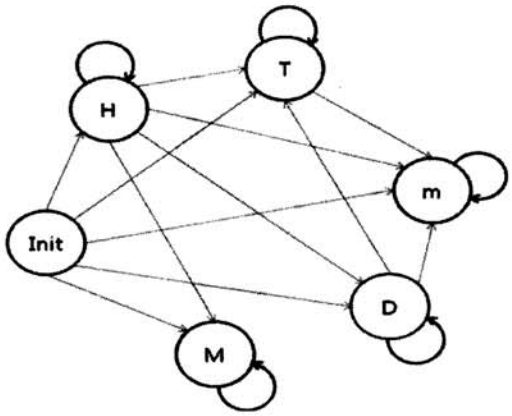
(그림 6) Hm 이벤트의 대체 이벤트(HD, HDm)



(그림 7) Tm 이벤트의 대체 이벤트(D, Dm, DTm)



(그림 8) HTm 이벤트의 대체 이벤트(HD, HDm, HDTm)



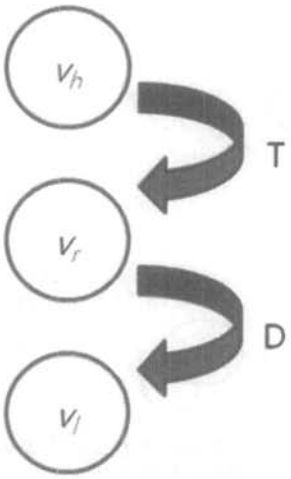
(그림 9) 연속 캐싱을 사용하는 트랜스코딩 프록시에서 D 이벤트를 추가한 이벤트 전이 그래프

17개의 이벤트 시퀀스가 발생 가능하다. 이러한 17개의 이벤트는 (그림 3)-(그림 8)과 같이 경우를 따지는 방법으로 확인할 수 있으며, (그림 9)에서 화살표 방향으로 전이해 봄으로써도 재확인할 수 있다.

요청한 버전 v_r에 대하여 DQS 정책을 적용할 수 있는 경우에 선택되는 낮은 품질의 버전을 v_l라고 하고, v_l이 전체 크기로 캐싱되어 있지 않을 때(partial caching), v_l의 나머지 부분(suffix)을 생성할 수 있는 v_r의 상위버전을 v_h라고 하면, 기존 이벤트들과 DQS 정책에서의 이벤트 사이의 관계를 <표 1>과 같이 정리할 수 있다. 표에서 $R(v)$는 버전 v를 캐싱하는 영역(range)를 나타낸다. 또한, 캐싱여부와 상관없음은 D.C(Don't Care)로 표현하였고, 일부가 캐싱되어 있음은 partial로 표시하였는데, 글자 크기가 크면 크기가 작은 것에 비해 더 많은 부분이 캐싱되어 있음을, 굵은 표시로 표시하면 더 많은 부분이 캐싱되어 있음을 나타내도록 하였다. 즉, <표 1>의 가장 마지막 줄의 기존의 HTm 이벤트 시퀀스가 HDTm 이벤트 시퀀스로 대체되는 상황을 표현할 때, v_h, v_r, v_l모두 부분을 캐싱하고 있지만, 그 영역의 상관관계가 각각$R(v_h) > R(v_r)$, $R(v_r) < R(v_l)$, $R(v_h) > R(v_l)$

〈표 1〉 기존 이벤트와 DQS 정책의 이벤트의 상관관계($R(v)$: 버전 v가 캐싱된 영역)

$R(v_h)$	$R(v_r)$	$R(v_l)$	DQS 이벤트	기존 이벤트
D.C	0	Full	D	T
Full	0	Partial	DT	T
D.C	0	Full	D	m
0	0	Partial	Dm	m
D.C	Partial	Full	HD	HT
Full	Partial	Partial	HDT	HT
D.C	Partial	Full	HD	Hm
Partial	Partial	Partial	HDm	Hm
D.C	0	Full	D	Tm
Partial	0	Partial	Dm	Tm
Partial	0	Partial	DTm	Tm
D.C	Partial	Full	HD	HTm
Partial	Partial	Partial	HDm	HTm
Partial	Partial	Partial	HDTm	HTm



(그림 10) T 이벤트와 D 이벤트의 관계

임을 나타내고 있다.

(그림 10)는 T 이벤트와 D 이벤트의 관계를 나타내고 있다. 사용자가 요청한 버전 v가 캐싱 되어 있지 않을 때, 트랜스코딩 프록시는 상위 버전인 v_h가 캐싱되어 있다면 트랜스코딩 연산을 통해 버전 v_h로부터 v_r를 생성하여 서비스를 제공한다. 만일, 트랜스코딩 프록시가 DQS 서비스 정책을 채택하고 DQS 시작 임계치를 넘은 상황이라면, 트랜스코딩 대신에 이미 캐싱되어 있는 낮은 품질 버전인 v_l를 통해서 서비스를 제공한다. 이때의 경우는 추가적인 트랜스코딩이 필요하지 않기 때문에, 시스템 부하를 줄일 수 있다.

한편, 사용자가 최상위 버전(v_0)을 요청한 경우에는 원본 서버로부터 페치(fetch)하여 서비스를 제공 하더라도 시스템 부하에 크게 영향을 미치지 않기 때문에, 차상위 버전(v_1)으로 서비스하는 것보다 원본 서버로부터 최상위 버전을 페치하여 서비스를 제공한다.

DQS 정책은 시스템의 가용 부하를 기준으로 시스템의 부하가 전혀 없을 때부터 적용할 수도 있고, 부하가 가용 부하를 넘어설 때만 수행할 수도 있으며, 부하 임계치(threshold)를 넘을 때만 제한적으로 사용할 수도 있다.

4. 성능 평가

본 논문에서는 제안한 DQS 정책의 성능을 측정하기 위해 [8, 10]에서와 유사한 시뮬레이션 환경을 이용하였다. 즉, 사용자 단말은 5개의 서로 다른 종류로 가정하였고 각각 15%, 20%, 30%, 20%, 15%의 비율로 서비스를 요청하며, 이때 단말은 각각 512, 256, 128, 64, 32Kbps의 상영속도(bitrate)를 지원한다. 사용자가 요청한 버전이 프록시에 저장되어 있지 않다면 이벤트의 종류에 따라 페칭 또는 트랜스코딩 연산을 통하여 사용자가 원하는 데이터를 전송한다. 트랜스코딩 지연시간은 트랜스코딩 연산이 발생할 때 연산이 시작된 이후에 처음으로 트랜스코딩 결과가 나오기 시작하기까지의 시간이며, 각 객체의 인기도는 Zipf 분포를 따르며 skew factor는 0.47로 가정하였다. 컨텐츠는 30초에서 12분 사이의 길이를 가지며, 시뮬레이션은 100,000개의 요청을 400 시뮬레이션 시간 동안 진행하도록 하였다. 원본서버로부터 트랜스코딩 프록시까지 데이터를 가져올 때의 지연시간은 지수분포를 따르며, 사용자 모두가 컨텐츠의 끝까지 보는 것은 아니기 때문에 지수분포로 종료 시간을 설정하였다. 프록시 서버의 저장공간은 128Kbps 컨텐츠의 전체 크기 합의 50% 의 크기를 사용하였다. 또한 세그먼트의 크기는 64Kbyte, 트랜스코딩 속도는 512Kbps로 설정하였다. 또한, 비교 대상으로는 TeC[6], VPF[8], 균등 세그먼트 기법(UNI)[10]으로 하고 DQS 정책은 기존의 모든 방법에 적용 가능하나, 본 논문에서는 균등 세그먼트 기법에 추가적으로 적용(UNI-DQS)하였다.

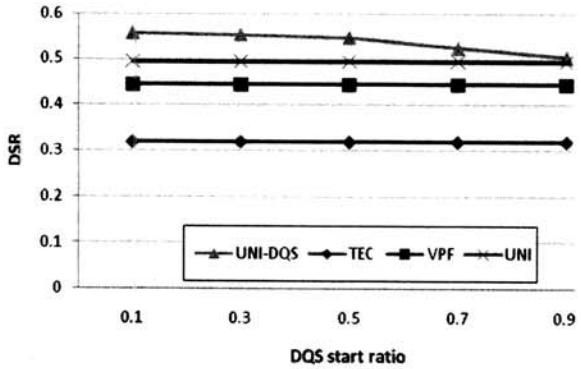
또한, 논문에서 설명한 DQS 정책을 실제 구현할 때 낮은 품질 버전이 복수 개 존재할 경우의 판단기준이 필요하다. 본 장에서는 복수 개 존재하는 낮은 품질 버전에서의 선택기준을 차 상위 버전(요청한 버전에 가장 근접하는 낮은 버전)을 선택하도록 하였다. 이러한 선택기준 방법은 여러 개 존재할 수 있으나, 버전의 종류가 매우 많아 요청한 버전이 일부영역에 대해 캐싱 되어 있거나 전혀 캐싱되어 있지 않고, 사용자가 요청한 버전보다 품질이 낮은 버전이 복수개 존재하며, 이들의 캐싱 범위가 요청한 버전의 캐싱 범위보다 클 때라는 매우 희박한 조건에 대해서만 성능차이가 발생하기 때문이다.

성능비교의 주요 기준은 DSR(Delay Saving Ratio)와 BHR(Byte Hit Ratio)이며, 트랜스코딩의 부하를 나타내도록 트랜스코딩 데이터 양을 추가적으로 측정하였다. 또한, 시스템 자원의 부족으로 지터가 많이 발생하는(10회 이상) 사용자의 수와 사용자의 품질 만족도를 측정하였다.

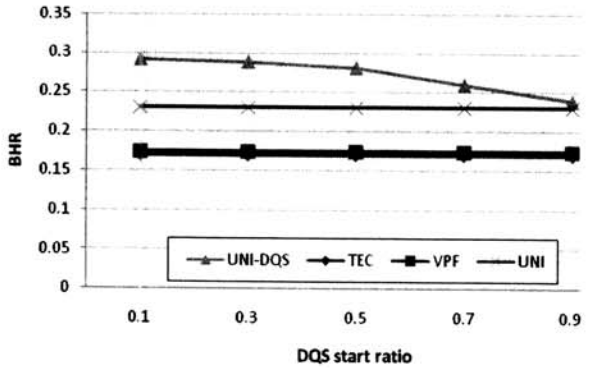
시스템에서는 항상 DQS 정책을 사용하는 것이 아니라, 시스템 부하가 적을 때는 트랜스코딩 연산을 수행하여 사용자의 만족도를 높이고, 부하의 임계치(threshold)를 넘어설 때는 트랜스코딩에 대해 D 이벤트로 대체 가능할 때 이벤트 대체가 발생한다. 따라서, 부하의 임계치 값을 어떻게 정할 것인가에 따라 성능이 달라진다. 아래의 실험결과 그래프에서 x축은 DQS 정책의 시작 임계치 값을 나타낸다. 그림들에서 UNI-DQS외의 방법들은 x축(DQS 시작 임계치)의 값에 상관없이 항상 같은 값을 가지는데, UNI-DQS 이외의 방법들은 시스템의 부하와 상관없이 동작하기 때문이다. 하지만, DQS 정책은 부하가 적을 때(0.1)부터 수행하게 되면, DSR, BHR, 트랜스코딩된 데이터 양, 지터 과발생 사용자 수에서 성능이 매우 우수하다. 하지만, 부하가 많을 때부터 수행하게 되면 UNI-DQS는 UNI와 거의 유사한 결과를 얻는다.

(그림 11)과 (그림 12)은 DQS 정책 시작 부하에 따른 DSR과 BHR의 값 변화를 보여준다. 그림에서 UNI-DQS 정책을 사용하게 되면 임계치가 0.1일때부터 적용한 경우가 0.9일때부터 적용한 경우에 비해 성능이 우수함을 알 수 있다.

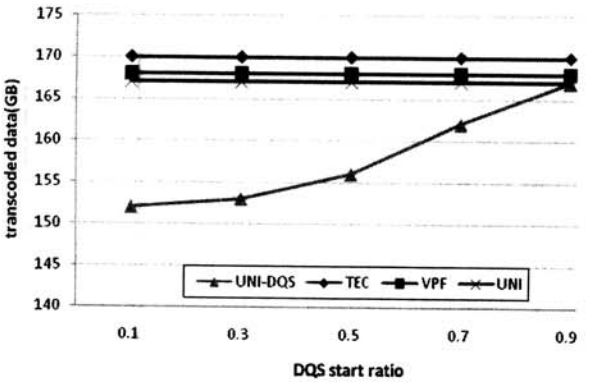
(그림 13)과 같이 DQS 정책을 사용하게 되면 트랜스코딩된 데이터 양을 줄일 수 있다. 특히 DQS 정책을 시스템 부



(그림 11) DQS 정책 시작 부하에 따른 DSR



(그림 12) DQS 정책 시작 부하에 따른 BHR



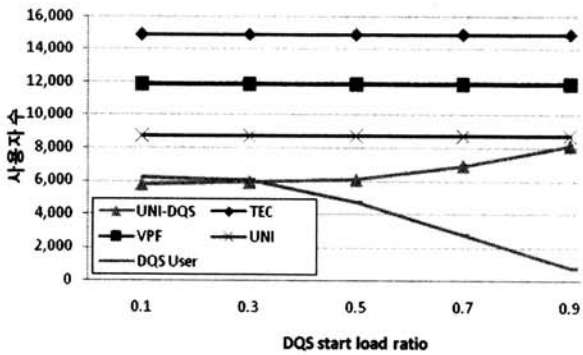
(그림 13) DQS 정책 시작 부하에 따른 트랜스코딩된 데이터 양

하가 적을 때부터 시작하게 되면 그 양을 월등히 줄일 수 있다.

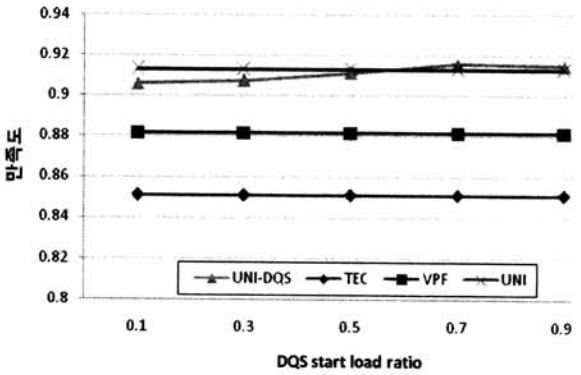
(그림 14)에서는 지터 과발생 사용자수(10회 이상 지터 발생 사용자)를 나타낸다. 또한 UNI-DQS 정책을 통하여, 실제 낮은 품질로 서비스를 받는 사용자의 수를 DQS User로 나타내었다. 그림과 같이 DQS 정책을 사용하면 지터 과발생 사용자 수를 줄일 수 있다.

(그림 11)에서 (그림 14)의 결과는 시스템 부하가 적을 때부터 DQS 정책을 사용하면 시스템의 성능 개선이 매우 크고, 시스템 부하가 클 때부터 DQS 정책을 적용하면 시스템의 성능 개선폭이 적어짐을 알 수 있다.

DQS 정책은 사용자가 요청한 품질을 만족시키지 않으면



(그림 14) DQS 정책 시작 부하에 따른 지터 과발생 사용자 수



(그림 15) DQS 정책 시작 부하에 따른 사용자 만족도

서 서비스를 제공하고자 하기 때문에 사용자의 만족도가 얼마나 떨어지는가가 중요한 요소중의 하나이다. 사용자의 만족도를 측정하는 방법은 주관적인 경우가 많으며, 객관적인 방법으로 사용하는 PSNR, Spectrum [11] 등도 본 실험에 부합하지는 않았다. 따라서, 본 논문에서는 서비스 만족도는 다음과 같이 계산하였다.

$$\text{만족도} = \frac{b_{ser}}{b_{req}} \quad (1)$$

즉, b_{ser}은 실제 서비스 받는 전송률을 나타내고, b_{req}는 사용자가 요청한 전송률을 나타낸다. 사용자 서비스 만족도는 사용자가 버전 0을 요청하여 버전 0으로 서비스 받으면 1, 버전 1로 서비스 받으면 0.5, 버전 2로 서비스 받으면 0.25 등이 되도록 계산하게 된다.

(그림 15)과 같이 논문에서 정의한 서비스 만족도로 계산하였을 때, DQS 정책을 사용하는 방법인 UNI-DQS는 제안 방법 UNI에 비해 거의 유사한 만족도를 가진다. 어떤 경우에 대해서는 오히려 UNI에 비해 우수한 경우도 있는데, 이는 보다 많은 사용자에게 서비스를 제공할 수 있기 때문이다.

실험과 같이 DQS 정책을 사용하면, DSR은 최대 11%, BHR는 최대 24% 향상되고, 트랜스코딩 되는 데이터는 최대 7% 줄이는 효과를 갖게 된다.

5. 결론 및 향후 연구

유비쿼터스 환경에서는 다양한 단말과 다양한 네트워크를 지원하는 사용자 적응형 서비스가 제공되어야 한다. 이러한 적응형 서비스를 기존의 프록시 서버에서 제공하면 사용자 적응 기능과 더불어 네트워크 대역폭의 절감과 사용자 지연 시간을 줄일 수 있다. 프록시 서버에서 적응형 서비스를 제공하기 위한 방법 중의 하나로 트랜스코딩 프록시에 관한 연구들이 진행되고 있는데, 트랜스코딩 프록시는 사용자 적응 서비스를 위해 고비용 연산인 트랜스코딩을 수행하는데 동시에 트랜스코딩 연산을 많이 수행하게 되면 일부 사용자들에게는 원활한 서비스를 제공할 수 없게 된다.

본 논문에서는 이러한 고비용 연산을 줄이기 위해 사용자의 품질 만족도를 제한적으로 희생하는 시스템 중심의 서비스 방법을 제안한다. 제안하는 시스템 중심의 서비스 방법을 낮은 품질 서비스(DQS: Degraded Quality Serivce)라고 정의하고, DQS의 구체적인 동작 상황을 설명하고, 이를 통하여 D 이벤트를 포함한 이벤트 전이 그래프로 총 17개의 이벤트 시퀀스를 정의하였다. 실험을 통하여 이러한 DQS 정책을 채택하는 방법과 채택하지 않는 방법의 성능을 비교한 결과, DQS 정책을 채택하면 시스템 측면의 자원을 절약할 수 있기 때문에 더 많은 사용자에게 서비스를 제공할 수 있는 반면, 사용자들의 품질 만족도는 떨어질 수 있음을 보였다. 이러한 DQS 정책은 사용자가 요청한 품질 기준을 반드시 만족해야 하는 경우에는 채택할 수 없겠지만, 그렇지 않은 경우는 시스템 자원을 적게 사용함으로써 보다 많은 사용자에게 서비스를 제공할 수 있다.

참 고 문 헌

[1] Aameek Singh, Abrishek Trivedi, Krithi Ramamritham, Prashant Shenoy, "PTC:Proxies that Transcode and Cache in Heterogeneous Web Client Environments," World Wide Web:Internat and Web Information Systems, Vol.7, No.1, Mar., 2004.

[2] Dongyu Liu, Songqing Chen, Bo Shen, "AMTrac:Adaptive Meta-caching for Transcoding," Proceedings of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2006), Newport, Rhode Island, May, 22-23, 2006.

[3] 서동만, 이좌형, 최면욱, 김윤, 정인범, "효과적인 트랜스코딩 부하 분산을 위한 자원 가중치 부하분산 정책", 정보과학회논문지: 컴퓨팅의 실제 제11권 제5호, 2005.10.

[4] Jiani Guo, Laxmi Bhuyan, "QoS Aware Job Scheduling in a Cluster-based Web Server for Multimedia Applications," 19th IEEE International Parallel and Distributed Processing Symposium(IPDPS05), 2005.

[5] Xueyan Tang, Fan Zhang, Samuel T. Chanson, "Streaming Media Caching Algorithms for Transcoding Proxies," Proceedings of the International Conference on Parallel Processing(ICPP02), 2002.

[6] Bo Shen, Sung-Ju Lee, Sujoy Basu, "Caching Strategies in Transcoding-Enabled Proxy Systems for Streaming Media Distribution Networks," IEEE Transactions on Multimedia, Vol.6, No.2, Apr., 2004

[7] Cheng-Yue Chang, Ming-Syan Chen, "On Exploring Aggregate Effect for Efficient Cache Replacement in Transcoding Proxies," IEEE Transactions on Parallel and Distributed Systems, Vol.14, No.6, Jun., 2003.

[8] Chi-Feng Kao, Chung-Nam Lee, "Aggregrate Profit-Based Caching Replacement Algorithms for Streaming Media Transcoding Proxy Systems," IEEE Transactions on Multimedia, Vol.9, No.2, Feb., 2007.

[9] YongJu Lee, YuHyeon Bak, OkGee Min, HagYoung Kim, CheolHoon Lee, "The PT-2 Caching Algorithm in the Transcoding Proxy Cluster to FacilitateAdaptive Content Delivery," International Workshop on Multimedia Content Analysis and Mining 2007(MCAM01), WeiHai, China, 2007.6.

[10] 박유현, 김학영, 김경석, "트랜스코딩 프록시에서 세그먼트 기반 캐시 교체 정책", 정보처리학회논문지A, 제15-A권 제 1호, 2008.2.

[11] Michael Zink, Jens Schmitt, Ralf Steinmetz, "Layer-Encoded Video in Scalable Adaptive Streaming," IEEE Transactions on Multimedia, Vol.7, No.1, Feb., 2005

박 유 현

e-mail : yhpark@deu.ac.kr

1996년 부산대학교 전자계산학과(학사)

1998년 부산대학교 전자계산학과(이학석사)

2000년 부산대학교 전자계산학과 박사수료

2000년 한국국방연구원(KIDA) 자원관리 연구부 연구원

2001년~2009년 2월 한국전자통신연구원 SW콘텐츠연구부문 인터넷플랫폼연구부 선임연구원

2009년 3월~현 재 동의대학교 컴퓨터소프트웨어공학과 전임강사

관심분야 : 트랜스코딩 프록시, 대규모 클러스터 시스템, 시스템 소프트웨어, VoD, 자료저장 시스템