무선 모바일 네트워크상에서 스트리밍 미디어 서비스를 위한 객체 버전 트랜스코딩

Object version Transcoding for Streaming Media Service in Wireless Mobile Networks

이종득*

Chong-Deuk Lee*

요 약

무선 모바일 네트워크상에서 트랜스코딩은 지연을 줄이고, 스트림의 처리 성능을 향상시키는 중요한 메커니즘이다. 그러나 무선 모바일 스트리밍 미디어 서비스는 제한된 네트워크 대역폭과 자원 등으로 인하여 혼잡, 간섭 및 지연등과 같은 문제점이 발생하고 있다. 간섭과 지연은 QoS를 떨어뜨릴 뿐만 아니라 스트리밍 미디어 서비스의 응답성을 떨어뜨린다. 본 논문에서는 객체 버전 트랜스코딩 기법을 제안한다. 제안된 기법은 객체 버전들을 분석하여 트랜스코딩 그래프를 구축한다. 스트리밍을 효율적으로 제어하기 위하여 참조율 기반 제어 함수를 이용하며, 지연 절약을 위해 MVDS(Multiple Version Delay Saving)를 측정한다. 시뮬레이션 결과 제안된 기법이 다른 비교 결과 기법들에 비해서 지연율, 캐시 히트율이 증가됨을 보인다.

Abstract

Transcoding in the wireless mobile network is an important mechanism that reduces the delay time and improves the stream processing capacity. Wireless mobile streaming media services, however, have such problems as congestion, interference and delay due to narrow network bandwidth and limited resources. These problems degrade not only Quality of Service (QoS) but also responsiveness of the streaming media service. To solve this problem, this paper proposes a object version transcoding method. The proposed method analyzes the object versions to construct the transcoding graph. This paper utilizes a reference rate-based control function for an efficient streaming, and measures MVDS(Multiple Version Delay Saving) for an efficient delay savings. The simulation results show that the proposed mechanism achieves improved performance in delay rate and cache hit rate compared with those of other existing methods.

Key words : Transcoding, Delay, Mobile Streaming, Object Version

I. 서 론

최근에 모바일 통신 기술 발전으로 인하여 사용자들은 노트북, PDA, 스마트 폰 등과 같은 모바일 기기들을 가지고서 언제 어디서나 인터넷에 접속하여 원하는 정보를 서비스 할 수 있게 되었다. 그러나 사용

* 전북대학교 공과대학 전자공학부(Div. of Electronic Engineering, Chonbuk National University)
· 제1저자 (First Author) : 이종득
· 투고일자 : 2011년 4월 6일
· 심사(수정)일자 : 2011년 4월 6일 (수정일자 : 2011년 6월 7일)
· 게재일자 : 2011년 6월 30일

자가 선호하는 차별화된 스트리밍 미디어 서비스를 제공받기 위해서는 트랜스코딩 기법이 제공되어야 한다[1],[2],[3],[4].

일반적으로 무선 모바일 네트워크상에서 적용되는 트랜스코딩 기법은 서버 기반 기법, 클라이언트 기반 기법, 그리고 프록시 기반 기법으로 분류된다[5],[6].

서버 기반 기법의 경우 웹 객체들의 다중 버전 트랜스코딩은 오프라인 상에서 수행되며, 이때 트랜스코드된 버전들은 서버 디스크에 저장된다. 서버 기반 기법의 장점은 클라이언트가 서버에 데이터를 요청할 때 트랜스코딩으로 인한 지연이 거의 발생하지 않는 다는 점이다. 그러나 지연을 줄이기 위하여 서버에서 객체의 여러 버전들을 독립적으로 저장하기 때문에 기억장소 낭비가 크다는 문제점이 있다. 또한 이 기법은 클라이언트의 요구사항을 능동적으로 반영하지 못하며, 저장되어 있는 각 버전들이 개별적으로 수행되는 문제점이 있다. 그리고 클라이언트 기반의 트랜스코딩은 모바일 클라이언트의 특성을 고려한 트랜스코딩 기법이다. 클라이언트 기반 기법의 장점은 원본 시멘틱 구조를 보존하여 프로토콜을 수행한다. 그러나 클라이언트 측에서 모바일 기기의 대역폭 제한과 에너지 소모 문제로 인하여 비용이 크게 발생하는 문제점을 가지고 있다. 이러한 문제를 해결하기 위하여 프록시 기반 트랜스코딩 기법들이 제안되고 있으며[7],[8], 프록시는 원본 서버와 모바일 클라이언트 사이에서 요청한 스트리밍 객체를 빠르게 트랜스코딩하는 기능을 수행한다. 뿐만 아니라 프록시는 요청한 객체를 클라이언트에게 전송하기 전에 적절한 버전으로 트랜스코딩하여 스트리밍 미디어 서비스를 빠르게 수행한다. 클라이언트가 요청한 객체를 트랜스코딩하는 프록시는 캐시 교체 알고리즘을 이용하여 스트리밍 미디어 서비스를 수행하며, 이를 위하여 coverage-기반 알고리즘과 demand-기반 알고리즘이 제안되었다 [1],[5]. coverage-기반 알고리즘은 객체의 원본 버전을 캐시하기 위한 알고리즘이며, 이에 반해서 demand-기반 알고리즘은 이미 트랜스코드된 객체 버전들을 캐시하기 위한 알고리즘이다. demand-기반 알고리즘은 가장 최근에 사용되지 않는 페이지를 제거하여 교체를 수행하는 demand-기반 LRU(Least Recently Used) 알고리즘과 참조 빈도율이 가장 적은 페이지와 교체를 수행하는 demand-기반 LFU(Least Frequency Used) 알고리즘으로 분류된다. 그러나 이들 알고리즘은 객체 원본과 이미 트랜스코드된 객체 버전만을 고려하여 캐시를 수행하기 때문에 캐시 용량, 객체 크기 속성 등으로 인해 인코딩율과 디코딩율의 불균형 문제가 발생한다. 따라서 본 논문에서는 이러한 문제를 해결하여 스트리밍 미디어 서비스를 효율적으로 수행하기 위한 새로운 객체 버전 트랜스코딩 기법을 제안한다. 제안된 알고리즘은 객체 버전 트랜스코딩을 위하여 캐시 용량, 버전 크기, 버전 제어 우선순위 등과 같은 여러 스트림 요소들을 반영하며, 그리고 새롭게 입력되는 객체 버전들을 트랜스코딩에 능동적으로 반영하기 위하여 참조율 기반 스트리밍 제어 기법 RRSC(Reference Rate-based Streaming Control)를 제안한다. 제안된 참조율 기반 스트리밍 제어 기법은 스트리밍을 수행할 객체 버전들에 대하여 참조 우선순위를 정하여 스트리밍을 제어하게 되며, 참조 우선순위가 빠른 객체 버전은 재전송으로 인한 지연을 줄임으로서 스트리밍 효과를 향상시키게 된다. 결과적으로 제안된 기법은 객체의 다중 버전들을 캐싱할 때 전체 캐싱 효과와 스트리밍 QoS 향상을 가져오도록 하는데 있다. 본 논문에서는 이벤트-지향의 시뮬레이션을 사용하여 제안된 기법의 성능을 평가하였으며, 시뮬레이션 결과 제안된 기법이 다른 비교 기법들에 비해서 지연 절약율과 캐시 히트율의 성능이 향상됨을 알게 되었다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해서 살펴보며, 3장에서는 제안된 객체 버전 트랜스코딩 모델에 대해서 살펴본다. 그리고 4장에서는 제안된 기법의 시뮬레이션 결과에 대해서 살펴보며, 끝으로 결론에 대해서 살펴본다.

Ⅱ. 관련연구

최근에 대부분의 트랜스코딩 프록시 구조는 유사한 네트워크 링크 및 디바이스 프로파일에 일치되는 모바일 클라이언트 중심으로 진화되고 있다. 무엇보

다 무선 모바일 네트워크 환경에서 스트리밍 QoS를 위해서는 객체 버전의 트랜스코딩이 고려되어야 한다[9],[10],[11]. 트랜스코딩이 수행된 객체 버전들은 보다 나은 스트리밍 미디어 서비스를 제공받을 수 있으나 트랜스코딩이 수행되지 않으면 대역폭 제한, 자원 제약, 트래픽 간섭 등으로 인하여 보다 나은 미디어 전송 서비스 품질을 제공 받을 수 없다[12],[13]. 따라서 많은 프록시 구조들은 스트리밍이 수행되는 미디어 클립 객체들을 스트리밍하기 위하여 트랜스코딩 기법을 적용하고 있다. 전통적인 트랜스코딩 프록시 기법들은 미디어 객체의 각 세그먼트 버전들을 분할하거나 객체의 여러 버전들을 통합하여 서비스를 수행하는 기법을 적용하고 있다. 이와 같은 방법을 위해서 Chang과 Chen[1]은 캐싱이 수행될 때 객체의 각기 다른 버전들을 분할하는 기법을 제안하였다. 그러나 이 기법은 웹 객체들의 분할에는 효율적이지만 미디어 객체의 경우에는 데이터 크기로 인한 스트리밍 제약이 발생하고 있다. Miao와 Ortega[5]는 상영 품질 (playback quality)을 향상시키기 위하여 선택적 캐싱 (selective caching)기법을 제안하였다. 이 기법은 프레임들이 전송율에 따라 캐시이 수행되도록 한 “중간 프레임 (intermediate frames)" 선택 기법이다. 그러나 이 기법에서 프록시는 시작 세그먼트 (initial segment) 이후의 프레임들은 직접 캐시하지 못하는 문제가 있다. Wu et al.[6]은 유연한 상영을 위해 동적 조절 비디오 버퍼 기법과 전송 스케줄링 기법을 제안하였다. 일반적으로 비디오 참조는 사용빈도가 높은 프레임과 사용빈도가 낮은 프레임으로 구분된다. 이와 같은 시나리오는 주로 라이브 뉴스 (live news)와 같은 인터넷 비디오에서 나타나고 있다.

Kao et al.[2]는 확장된 가중치 트랜스코딩 그래프를 이용하여 범용 객체 이득을 계산하는 함수를 제안하였다. 이 기법은 스트림을 수행할 때 가중치를 식별하여 트랜스코딩이 수행되도록 하였으며, 이득함수에 기반을 두고서 캐시 교체가 수행되도록 하였다. 그러나 이 기법은 사용빈도가 높은 객체들과 그렇지 않는 객체 버전들을 식별하기 어렵다는 문제점이 있다.

따라서 본 논문에서는 이러한 문제를 해결하고 스트리밍 미디어 서비스의 성능을 최적화하기 위한 객체 버전 트랜스코딩 기법을 제안한다.

Ⅲ. 객체버전 트랜스코딩 메커니즘

이 장에서는 제안된 객체 버전 트랜스코딩을 위한 트랜스코딩 그래프, 참조율 기반 제어함수, 그리고 버전 스트리밍 제어에 대해서 살펴본다.

3-1 트랜스코딩 그래프

트랜스코딩이란 멀티미디어 객체를 이미 코딩된 신호를 다른 신호로 바꾸기 위한 변환과정이다. 트랜스코딩이 수행되는 원본 객체는 완전한 상태의 미디어 콘텐츠 정보이며, 원본 객체는 원본 버전 또는 가장 상세한 객체 버전과 대응된다[2],[4]. 상세한 객체 버전은 비교적 덜 상세한 객체 버전으로 트랜스코드되며, 본 논문에서는 이러한 객체 버전을 트랜스코드된 버전이라 정의한다. 일반적으로 객체의 손실이 없을 경우 각 객체는 n개의 버전들로 표시되며, 여기서 V1 즉 버전1은 객체의 원본 버전이다. 그리고 Vn 즉 버전n은 더 이상 트랜스코드 되지 않는 버전이다. 이들 중간 버전들을 차례로 V2, ..., Vn-1 이라 하고, $i < j$일 때 각 i, j에 대해서 Vi는 Vj보다 더 상세한 버전이라 한다. 그러나 트랜스코딩을 수행하는 과정에서 Vi가 Vj에 대한 완전한 트랜스코딩 정보를 가지지 않고, $i < j$이면 이 경우 모든 Vi는 Vj로 트랜스코드 되지 않는다. 따라서 객체 버전 트랜스코딩을 위해서 프록시는 트랜스코딩을 수행할 객체의 트랜스코드 관계와 참조율을 미리 알아야 하며, 참조율에 기반한 트랜스코딩 그래프 G_i는 다음과 같이 정의된다.

(정의1) 참조율을 가진 트랜스코딩 그래프 G_i는 참조율 w_i를 가진 방향성그래프이다. 여기서 G_i는 트랜스코드가 가능한 객체i의 버전들 간의 트랜스코딩 관계를 나타낸다. 각 정점(vertex) $v \in V[G_i]$에 대해서 v는 트랜스코드가 가능한 객체i의 버전이다. 만일 방향성 에지 $(u,v) \in E[G_i]$가 존재하면 객체i의 버전u는 버전 v로 트랜스코드된다. 버전u에서

버전 v까지의 트랜스코딩 비용은 참조율 $w_i(u,v)$로 나타내며, 이것은 버전u에서 버전 v까지의 에지의 참조율이다.

실제로 참조율 기반의 트랜스코딩 그래프는 트랜스코딩 수행 여부에 따라 각 객체 버전에 대해서 참조율이 적용되며, 트랜스코딩이 수행되지 않는 버전은 참조율이 0이 된다. 그림1은 참조율에 기반한 그래프이며, 그림1에서 원본 버전 V1은 비교적 덜 상세한 버전인 V2, V3, V4로 트랜스코드 된다.

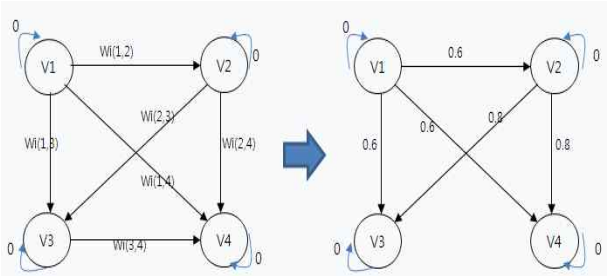

그림 1. 참조율 기반의 트랜스코딩 그래프
Fig.1 Reference-based Transcoding Graph

그림1에서 V1, V2, V3, V4는 트랜스코딩이 수행되지 않는 자기 자신의 버전이며, 이때 객체 버전에 대한 콘텐트 정보가 제공되지 않으면 객체 버전은 트랜스코딩을 수행하지 않게 된다.

3-2 참조율 기반 제어 함수

제안된 알고리즘에서는 참조율 기반 제어함수를 이용하여 버전 제어 우선순위를 결정한다. 버전 제어는 단일 버전 제어와 다중 버전 제어로 구분하여 수행되며, 이것은 캐싱을 효율적으로 수행하기 위한 것이다.

버전 제어 정의를 위해 o_i^j 를 객체i의 버전j라 하자. 객체들의 서로 다른 버전들에 대한 참조율은 통계적으로 서로 독립이라 가정하며, r_i^j로 표현한다. 여기서 r_i^j는 객체i의 버전j에 대한 평균 참조율이다. 객체i를 원본 서버에서 프록시 서버로 패치하는데 걸리는 평균 지연은 d_i로 표현한다. 여기서 d_i는 객체를 패치하는데 걸리는 지연시간이다.

객체i의 버전j에 대한 크기는 s_i^j로 표현한다. 객체i에 대응되는 가중치를 가진 트랜스코딩 그래프는 G_i로 표현하며, 객체i의 버전x에서 버전y까지를 트랜스코딩하는데 걸리는 지연은 E[G_i]의 경우 에지 (x,y)의 참조 가중치 w_i로 표현한다. 따라서 E[G_i]에서 에지(x,y)의 참조 가중치는 $w_i(x,y)$이다. 따라서 버전 제어 파라미터 리스트는 표1과 같다.

표 1. 버전 제어 파라미터 리스트
Table 1. Version Control Parameter List

파라미터	의미
o_i^j	객체i의 버전j
r_i^j	객체i의 버전j에 대한 평균 참조율
d_i	콘텐츠서버에서 트랜스코딩 프록시로 객체i를 패치하는데 걸리는 지연시간
s_i^j	버전j의 크기
G_i	참조율 기반의 트랜스코딩 그래프
$w_i(x,y)$	G_i에서 참조 가중치

클라이언트 관점에서 볼 때 최적의 스트리밍 서비스가 수행되기 위해서는 객체 버전에 대한 참조율이 고려되어야 하며, 참조율이 클수록 높은 응답성이 보장되며 지연이 최소화되게 된다. 이를 위해서 객체 버전에 대한 지연 측정은 매우 중요하며, 객체 버전들 중에서 객체i의 버전j에 대한 단일 버전 지연 절약 SVDS (Single Version Delay Saving) 측정은 식 (1)과 같이 정의된다.

(정의2) $\mathrm{SVD}(o_i^j)=$

$$\sum_{(j,x)\in E[G_i]} r_i^x \times\{d_i+\omega_i(1,x)-\omega_i(j,x)\} \quad (1)$$

식(1)에서 각각의 $(j,x)\in$E[G_i]는 버전j에서 버전x까지의 각기 트랜스코드가 가능한 버전을 나타낸다. 식 $d_i+w_i(1,x)$는 o_i^j가 참조되지 않았을 때의 지연이다. 즉 원본 콘텐츠 서버에서 객체i를 패치하는 데 걸리는 시간과 원본 객체 버전에서 버전x까지 트랜스코딩하는 데 걸리는 시간을 합한 것이다. 그리

고 $w_i(j,x)$는 o_i^j가 참조되었을 때의 지연을 나타낸다. 따라서 $\{d_i+w_i(1,x)-w_i(j,x)\}$는 단일 버전 o_i^x에 대한 지연 절약을 수행하게 된다.

단일 버전에 대한 참조는 응답성과 지연이 어느 정도 보장되지만 동시에 여러 버전들이 요청될 때에는 응답성이 떨어지게 되어 지연이 증가하게 된다. 이를 해결하기 위해서 본 논문에서는 다중 버전에 대한 응답성을 보장하고 지연 절약을 수행하기 위한 다중 버전 지연 절약 MVDS (Multiple Version Delay Saving)을 측정하며, 식 (2)와 같이 정의된다.

(정의3) $$\mathrm{MVDS}(o_i^{j1},o_i^{j2},...,o_i^{jk})=\sum_{v\in V[G']}\sum_{(v,x)\in E[G']} r_i^x\times\{d_i+\omega_i(1,x)-w_i(v,x)\} \quad (2)$$

여기서 G'는 (G, w, $\{j1,j2,\ldots,jk\}$)에 의해서 생성된 서브그래프이다.

그리고 객체 버전 $(o_i^{j1},o_i^{j2},...,o_i^{jk})$가 트랜스코딩 프록시에 캐시될 때에는 이미 캐시된 객체 버전과 다음에 캐시될 객체 버전들을 구분하여 참조율에 따른 캐싱 이득이 측정되어야 하며, 다음에 캐시될 객체 버전들에 대한 이득 측정 PM (Profit Metric)은 식 (3)과 같이 정의된다.

(정의4) $$\mathrm{PM}(o_i^j|\ o_i^{j1},o_i^{j2},...,o_i^{jk})=SVD(o_i^{j1},o_i^{j2},...,o_i^{jk})-\overline{SVD}(o_i^{j1},o_i^{j2},...,o_i^{jk}) \quad (3)$$

트랜스코딩 프록시는 동시에 2개 이상의 객체의 버전들을 참조하고 캐시할 수 있으며 이러한 과정을 수행할 때 단일 버전에 대한 응답성이 보장되고 트랜스코딩 프록시에서 전체적인 캐싱 이득도 증가하게 된다. 그러나 1장에서 기술된바와 같이 전체적인 캐싱 이득은 객체의 서로 다른 버전들 간의 트랜스코딩 관계성에 의해서 결정된다. 따라서 객체의 다중 버전들을 동시에 참조할 때 참조율에 따른 지연을 최소화하고 캐싱 이득을 극대화하기 위한 과정은 다음과 같이 수행된다.

```
procedure profit calculation
input: G, w, o_i^j
output: maximum profit
 i←0;j←0
 // 객체 i와 버전 j에 대하여 초기화
  for each vertex u∈o_i^j
    calculate MVDS(o_i^j1,o_i^j2,...,o_i^jk)
  for each vertex v∈V[G]
      do for each (x,v)∈E[G] and
      MVDS(o_i^j1,o_i^j2,...,o_i^jk)
        do find the maximum w(x,v)
  P←P∪(x,v)
  // P는 반환될 최대 이득이다.
 return G'(V[G], P)
```

procedure profit calculation의 입력은 G, w, o_i^j의 3개의 요소로 구성되어 있으며, 여기서 G는 참조율 기반의 트랜스코딩 그래프이다. 그리고 w는 G에서 에지들의 참조 가중치이며, o_i^j는 최대 이득을 계산하기 위한 객체 버전들이다.

객체 버전들에 대한 캐싱 이득을 계산하기 위해서는 객체 크기를 반영해야 하며, 객체 크기를 반영하기 위해서는 캐시 용량에 따른 객체의 크기가 분할되어야 한다. 객체 분할 크기는 []에 기반하여 분할되었으며, 객체 분할은 스트리밍 미디어 서비스 성능의 중요한 척도로 작용된다. 따라서 제안된 알고리즘은 객체 분할에 기반하여 최대 이득이 측정되도록 하였다.

3-3 스트리밍 제어

본 논문에서 수행되는 스트리밍 제어 사상은 참조율이 낮은 객체 버전을 제어하여 차별화된 스트리밍 서비스를 보장하기 위한 것이며, 제안된 스트리밍 제어 사상은 캐시 속성에 기반을 둔다. 따라서 차별화된 스트리밍 서비스를 보장하기 위한 스트리밍 제어 사상은 캐시 용량, 객체의 크기, 그리고 참조율에 의해 제약을 받으며, 제약조건 C가 $c\in C$ 일 때 스트리밍 제어 사상은 $disjunction(\cup)$사상,

$conjunction(\cap)$사상, 그리고 필터링 사상으로 구분하여 수행된다.

ⅰ) $disjunction(\cup)$사상

$disjunction(\cup)$사상은 캐시용량, 객체의 크기를 모두 고려한 사상으로서 $Max-Min$ 관계를 가진 객체 버전들을 제어하기 위한 정책이다. 캐시 용량을 고려한 스트리밍 제어 사상은$M_{disjunct}^{C,C}$ 으로 표현하고, 객체의 크기를 고려한 스트리밍 제어 사상은 $M_{disjunct}^{s}$ 으로 표현한다. 따라서 $disjunction(\cup)$에 의한 사상은 다음과 같이 정의된다.

(정의5) $M_{disjunct}^{(C,C)\cup s}$: $\forall c,c' \in C$이고 c는 $c \cup c'$이다.

여기서 C,C는 Cache Capacity이며, s는 객체의 크기이다.

ⅱ) $conjunction(\cap)$사상

$conjunction(\cap)$사상은 캐시용량, 객체의 크기, 그리고 참조율을 고려한 사상으로서 $Min-Max$ 관계를 가진 객체 버전들을 제어하기 위한 전략이다. 캐시 용량을 고려한 제어 사상은 $M_{conjunction}^{C,C}$, 객체의 크기를 고려한 제어 사상은 $M_{conjunction}^{s}$, 그리고 참조율을 고려한 제어 사상은 $M_{conjunction}^{\omega}$ 으로 표현한다.

따라서 $conjunction(\cap)$에 의한 사상은 다음과 같이 정의된다.

(정의6) $M_{conjunction}^{(C,C')\cap s\cap \omega}$: $\forall c,c' \in C$이고 c는 $c \cap c'$이다.

여기서 $\forall c,c' \in C$, $\varnothing$은 c의 $null$이다.

ⅲ) $filtering$ 사상

$filtering$ 사상은 $disjunction(\cup)$사상과 $conjunction(\cap)$사상을 만족하지 않는 객체 버전들을 제어하기 위한 전략이다. $filtering$ 사상은 $disjunction(\cup)$사상과 $conjunction(\cap)$사상을 통해서 수행되며, $filtering$을 위한 사상은 $f_i \in F$일 때 $M_{filtering}^{\omega}$로 표현한다. 따라서 $filtering$을 위한 제어 사상 $M_{filtering}^{\omega}$은 다음과 같이 정의된다.

(정의7) $M_{filtering}^{\omega}$={ $M_{filtering}^{\omega}(O)$ | $(o_i^j | o_i^{j1},o_i^{j2},...,o_i^{jk}) \geq \omega$}

예를 들어 표2에서 미디어 객체 O1, O2, O3에 대해서 $M_{filtering}^{0.1\cup 0.4}(o)$을 수행하면 O1=$\{o_1^1,o_1^3\}$, O2=$\{o_2^2,o_2^4\}$, O3=$\{o_3^3,o_3^4\}$의 객체 버전이 제어된다.

표 2. 객체 버전 참조 테이블
Table 2. Object version Reference Table

O / o_i^j	o_1^1	o_1^2	o_1^3	o_1^4
O_1	0.3	0.7	0.2	

O / o_i^j	o_2^1	o_2^2	o_2^3	o_2^4
O_2	0.6	0.1	0.8	0.4

O / o_i^j	o_3^1	o_3^2	o_3^3	o_3^4
O_3	0.5	0.9	0.3	0.2

Ⅳ. 시뮬레이션 분석

이 장에서는 이벤트-지향 시뮬레이션(event-driven simulation)을 수행하여 제안된 기법의 성능을 비교분석한다. 이벤트 지향 시뮬레이션의 목적은 파라미터들을 변화시키면서 성능을 평가하기 위한 것이다. 시뮬레이션 분석을 위해 4.1절에서는 시뮬레이션 환경에 대해서 살펴보며, 4.2절에서는 시뮬레이션 결과에 대해서 살펴본다.

4-1 시뮬레이션 환경

시뮬레이션을 위한 텍스트 객체와 이미지 객체의 크기는 각각 8Kbyte와 2Mbyte 이내로 제한하였으며 Pareto 분포[2]를 이용하였다. 트랜스코딩 지연 측정은 트랜스코딩 참조율과 객체의 크기를 이용하였으며, 미디어 객체들의 참조율을 위한 중요도는 Zipf-like 분포[1]를 이용하였다. 모든 미디어 객체의

버전과 속성을 설정한 후 가장 최근의 작업 부하를 반영하여 클라이언트의 작업부하를 차례로 생성하였다. 각각 생성된 작업부하는 50,000개의 사용자 접근으로 구성되었다. 접근한 시퀀스에 대한 시퀀스의 길이는 3.5로 제한하였으며, 접근한 시퀀스들이 응답하는 응답 시간은 0.2초 이내로 설정하였다. 클라이언트가 트랜스코딩 프록시 서버에 데이터 요청을 한 후 데이터를 스트리밍하는 데 걸리는 시간은 5초 이내로 설정하였다. 스트리밍을 위한 객체의 트랜스코딩 율은 초당 12byte로 설정하였으며, 캐시의 기본 용량은 0.05*($\sum$객체크기)로 설정하였다. 콘텐츠 서버로부터 미디어 객체들을 패치하는 데 걸리는 지연은 지수분포를 이용하였으며, 콘텐츠 서버로부터 미디어 객체를 패치하는 데 걸리는 시간과 객체 크기와의 상관관계는 거의 없다고 가정하였다.

4-2 시뮬레이션 결과

본 논문에서는 성능 평가를 위해서 객체 수, 객체 크기, 시퀀스 길이, 캐시용량, 그리고 참조가중치 등의 파라미터를 변화시켜 가면서 시뮬레이션을 수행하였다. 시뮬레이션에서 사용된 주요 성능 척도는 캐시 제어율, 지연 절약율, 그리고 스트림 제어율이다. 캐시 제어율이란 적합도를 만족하는 스트림 데이터들이 프록시 버퍼 캐시에 얼마나 만족되는지를 나타내는 비율이며, 지연 절약율은 버퍼 캐시에서 스트림 데이터들이 히트되어 대기 지연이 적게 발생하는 비율을 말한다. 그리고 스트림 제어율은 스트림 데이터들이 손실이 발생되지 않고 버퍼 캐시에 히트되는 비율을 말한다. 본 논문에서는 이들 성능 척도에 기반을 두고서 시뮬레이션을 수행하였으며, 성능 비교는 coverage-기반 기법과 demand-기반 기법으로 구분하여 수행하였다.

i) 참조 가중치에 의한 캐시 제어율

첫 번째 성능평가는 참조 가중치를 적용한 캐시 제어율이다. 참조 가중치 ω가 0.5 이하일 때는 버퍼 캐시 제어율에 영향을 미치게 된다. 이에 대한 시뮬레이션 결과는 그림2와 같다.

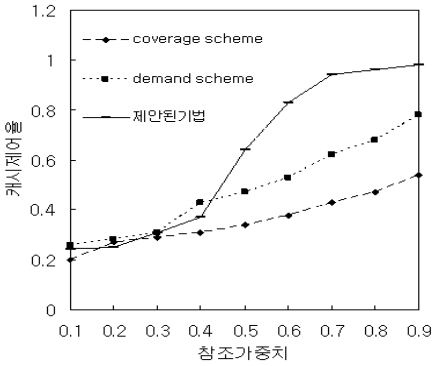

그림 2. 버퍼캐시 제어율
Fig. 2. Buffer Cache Control Rate

그림2에서 보듯이 제안된 기법은 비교적 우수한 coverage 기법보다 버퍼 캐시 제어성능이 향상되었음을 알 수 가 있다. 이것은 제안된 기법에서는 스트림 데이터들에 대해서 트랜스코딩을 수행했기 때문이다. 따라서 객체 버전들을 버퍼 캐시에 적합하도록 트랜스코딩 할 때 버퍼 캐시가 효율적으로 제어됨을 알 수 있다. 그러나 참조율 관점에서 볼 때 참조 가중치 ω는 버퍼 캐시에 영향을 미치게 된다. 만일 참조 가중치 ω값이 ω<0.7일 때는 콘텐츠 서버로부터의 패치 지연이 크게 발생하기 때문에 버퍼캐시의 성능에 영향을 미치게 된다. 본 논문에서는 스트림 데이터가 너무 작거나 캐시 용량을 초과하는 스트림 객체 버전들에 대해서 스트림 제어 사상을 적용하여 버퍼 캐시가 제어되도록 하였다.

ii) 객체 수에 따른 평균 지연 절약률

두 번째 성능평가는 객체 수를 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000으로 점차 증가시켜 가면서 평균 지연 절약율을 평가하였다. 이에 대한 시뮬레이션 결과는 그림3과 같다.

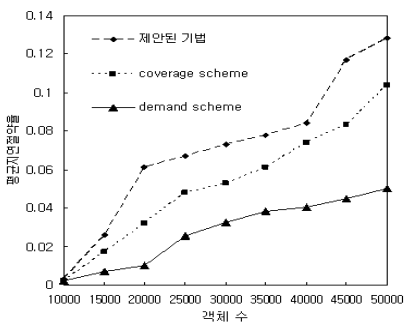

그림 3. 평균 지연 절약율
Fig. 3. Average Delay Saving Rate

그림3에서 보듯이 객체 수가 증가할 때 제안된 기법의 지연 절약율은 비교적 높게 나타났음을 알 수 있다. 이것은 참조 가중치 ω와 트랜스코딩이 스트림에 영향을 미치기 때문이다. 따라서 제안된 기법은 각 객체 버전들에 대한 참조 가중치 ω와 트랜스코딩이 반영되기 때문에 객체 수가 증가해도 다른 기법들에 비해서 성능이 향상되는 결과를 가져온다.

iii) 캐시 용량에 따른 스트림 제어율

세 번째 성능평가는 캐시 용량에 따른 스트림 제어율이다. 스트림 제어율 평가를 위해 캐시 용량을 변화시켜가면서 시뮬레이션을 수행하였다. 이에 대한 분석 결과는 그림4와 같다.

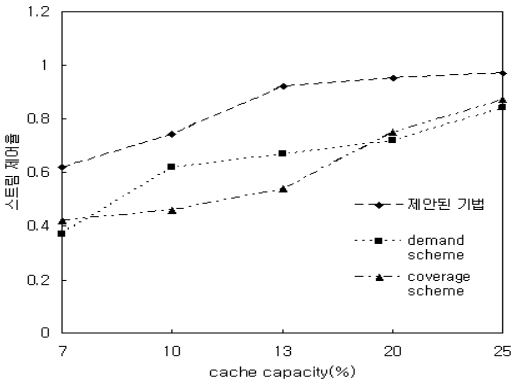

그림 4. 캐시 용량에 따른 스트림 제어율
Fig. 4. Stream Control Rate by Cache Capacity

그림4에서 보듯이 제안된 기법은 스트림 제어율 관점에서 볼 때 다른 기법들에 비해서 성능이 우수함을 알 수 있다. 특히 비교적 성능이 우수한 coverage 기법과 비교해 볼 때 약 20%의 성능이 향상되었다. coverage 기법과 demand 기법의 성능이 상대적으로 낮은 이유는 객체 버전에 대한 이득 값을 반영하지 않았기 때문이다. 특히 스트리밍 미디어 객체는 웹 객체와는 달리 프록시 서버에서 더 큰 지연 문제가 발생된다. 그러나 제안된 기법은 객체 버전들에 대한 이득 측정값 PM을 스트리밍에 반영하였기 때문에 프록시 서버에서 크기가 큰 객체 버전이 새롭게 입력되어도 스트림 제어를 효율적으로 수행하게 된다.

Ⅴ. 결 론

최근에 무선 모바일 네트워크상에서 캐싱 성능과 스트리밍 QoS 성능을 향상시키기 위하여 트랜스코딩 기법에 대한 많은 연구가 수행되고 있다. 그러나 무선 모바일 스트리밍 미디어 서비스를 위한 트랜스코딩은 네트워크 대역폭 제한과 자원 제약 등으로 인하여 스트리밍 성능에 많은 문제점이 발생하고 있다. 이러한 문제를 해결하기 위하여 본 논문에서는 객체 버전 트랜스코딩 기법을 제안하였다. 제안된 기법은 효율적인 스트리밍 미디어 서비스를 위해서 객체 버전들에 대한 참조율을 측정하였으며 측정된 참조율에 따라 트랜스코딩 그래프가 구축되도록 하였다. 그리고 트랜스코딩 수행여부에 따른 지연절약을 측정하기 위하여 지연절약을 단일 버전 지연절약과 다중 버전 지연 절약으로 구분하여 측정하였다. 또한 참조 관련성이 낮은 객체 버전들을 제어하기 위하여 스트리밍 제어 사상을 수행하였다. 우리는 실험 결과를 통하여 제안된 기법의 성능이 다른 기법들에 비해서 효율적이었음을 알 수 있었다.

참 고 문 헌

[1] Cheng-Yue Chang and Ming-Syan Chen, "On Exploring Aggregate Effect for Efficient Cache Replacement in Transcoding Proxies," *IEEE TRANSACTIONS ON PARELLEL DISTRIBUTED SYSTEM,* VOL. 14, NO. 6, pp. 611-624, 2003.

[2] Chi-Feng Kao and Chung-Nan Lee, "Aggregate Profit-Based Caching Replacement Algorithms for Streaming Media Transcoding Proxy Systems," *IEEE TRANSACTIONS ON MULTIMEDIA,* VOL. 9, NO. 2, pp. 221-230, 2007.

[3] Songqing Chen, Bo Shen, Susie Wee, and Xiaodong Zhang, "Segment-Based Streaming Media Proxy: Modeling and Optimization," *IEEE TRANSACTIONS ON MULTIMEDIA,* VOL. 8, NO. 2, pp. 243-256, 2006.

[4] Kuei-Chung Chang, Tien-Fu Chen, "Efficient Segment-based Video Transcoding Proxy for Mobile Multimedia Services," *Journal of Systems*

Architecture 53, pp. 833-845, 2007.

[5] Zhourong Miao, Antonio Ortega, "Proxy caching for Efficient Video Services over the Internet," *in:Packet Video Workshop,* pp. 1-21, 1999.

[6] Dapeng Wu, Yiewei Thomas Hou, Wenwu Zhu, Ya-Qin Zhang and Jon M. Peha, "Sreaming Video over the Internet: Approachs and Directions," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, VOL. 11, NO. 3, PP. 282-300, 2001.

[7] A. Maheshwari, A. Sharma, K. Ramamrithan, and P. Shenoy, "Transquid: Transcoding and Caching Proxy for Hetereogeneous e-commerce Environments," *in Proc. IEEE INFOCOM 2002*, San Jose, CA, pp. 50-59, 2002.

[8] Danjue Li, Chen-Nee Chuah, Gene Cheung and S. J. Ben Yoo, "MUVIS : Multi-source Video Streaming Service over WLANs", *Journal of Communication and Networks(JCN)*, vol.7, pp. 144-156, 2005.

[9] C. D LEE, T. W. JEONG, "Fuzzy Filtering-based Segment Grouping for User-Centered Multimedia Streaming Service in P2P Distribution Mobile Networks," *Journal of Internet Technology,* Vol. 11, No. 5, pp. 651-658, 2010.

[10] C. D. LEE, "Proxy Caching Grouping by Partition and Mapping for Distributed Multimedia Streaming Service," *KIIS,* Vol 19, No, 1, pp. 40-47, 2009.

[11] C. D LEE, T. W. JEONG, J. Y. AHN, "A Streaming Service based on Partition and Mapping of Media Block," *ISME2009, The 6th International Symposium on Management Engineering*, 5-7 August, Dalian, China, CD, pp. 1-6, 2009.

[12] 이종득, “무선 네트워크상에서 멀티미디어 스트리밍 최적화를 위한 전송율 기반의 오버헤드 모니터링”, *한국항행학회 논문지,* Vol. 14-3, pp. 358-366, June, 2010.

[13] 이종득, “무선 애드혹 네트워크상에서 라우팅 성능향상을 위한 퍼지 적합도 기반 클러스터링,” *한국항행학회 논문지,* Vol. 14-4, pp. 495-503, August, 2010.

이 종 득 (李鍾得)

1983년 2월 : 전북대학교 컴퓨터과학과(이학사)
1989년 2월 : 전북대학교 컴퓨터과학과(이학석사)
1998년 2월 : 전북대학교 컴퓨터과학과(이학박사)
1992년 3월~2002년 2월 : 서남대학교 컴퓨터통신학과 교수
2002년 2월~2011년 6월 현재 : 전북대학교 전자공학부 교수
관심분야 : 무선 모바일 네트워크, 무선센서 네트워크, MIMO, 유비쿼터스 통신, 등