다중 디스크립션 비디오 스트리밍을 위한 협업 프록시 구조 설계

이승은°, 한종욱, 한동윤, 박유현‡, 김경석†
부산대학교 컴퓨터공학과°, 한국전자통신연구원‡, 부산대학교 정보컴퓨터공학부†
{selee°, jwhan, dyhan, gimgs0†}@asadal.pusan.ac.kr, bakyh‡@etri.re.kr

Cooperative Proxy Architectural Design for Multiple Description Video Streaming

Seung-eun Lee°, Jong-wook Han, Dong-yun Han, Yuhyeon Bak‡, Kyong-sok Kim†
Dept. of Computer Engineering°. Electronics and Telecommunications Research Institute‡,
Division of Computer Science and Engineering. Pusan National University†

요 약

최근 많은 사용자들이 인터넷 상에서 비디오나 오디오 같은 규모가 큰 미디어 콘텐츠(Media contents)의 정보 전달을 효율적으로 제공 받기를 원하고 있다. 미디어 콘텐츠를 스트리밍 하는 것은 인터넷 트래픽의 아주 큰 부분을 차지하기 때문에, 우리는 클러스터형 협업 프록시를 기반으로 한 다중 디스크립션 비디오 스트리밍 구조를 제안하여 네트워크와 서버의 부하를 클러스터된 일반 프록시들에게 균등하게 분산 시킨다. MDC(Multiple Description Coding)[1]를 적용하여 캐싱 공간(Caching space)을 효율적으로 사용하며 다중 세션을 통해 디스크립션을 서비스함으로써 스트리밍 서비스 중인 프록시 또는 전체 데이터를 저장하고 하는 서버에 문제가 생기더라도 끊어지지 않는 계속적인 서비스가 가능하다. 또한 일반 프록시들의 정보를 관리하는 디스패처가 사용자의 환경에 따라 적응적인 서비스를 가능하게 한다.

1. 서 론

최근 인터넷 접속 환경의 발달과 시스템 및 네트워크 성능 향상의 영향으로 많은 사용자들은 텍스트나 이미지 같은 단순한 정보 요구뿐만 아니라 큰 용량과 많은 대역폭 사용 등의 특징을 가진 VoD(Video-on-Demand : 주문형 비디오) 서비스를 요구하고 있다. 사용자들은 서비스 요청에 대한 빠른 응답으로 서비스를 받기를 원한다. 이러한 초기 접근 시간과 서버/네트워크 부하를 줄이기 위한 효율적인 방안으로 클라이언트들 가까이에 있는 프록시들에 빈번히 사용되는 데이터를 캐시(cache)하는 것이 있다.

기존 프록시들은 웹 페이지들을 캐시하기 위해 효과적이다. 하지만 일반적으로 비디오 파일은 웹 페이지보다 수백만 배 더 크기 때문에 하나의 프록시에 캐시되는 비디오 파일은 매우 제한되어 몇 개만 저장하더라도 캐시 공간을 모두 소비하게 된다. 그렇기 때문에 비디오나 오디오 파일과 같은 스트리밍 서비스를 위한 멀티미디어 프록시에 대한 연구들이 진행되어 왔다.

초기의 멀티미디어 프록시 연구는 주로 다수의 같은 종류(네트워크 모델, 장치 구성 등)의 클라이언트만을 대상으로 서비스 하였다. 같은 종류의 클라이언트에 대한 스트림(Stream) 캐시를 위해, 캐시 할 부분에 따라 기존 알고리즘을 Sliding-Interval caching[3], Prefix caching[4,5], Segment caching[6,7,8,9] 그리고 Rate-Split caching[10]과 같은 연구들이 나왔다. 최근에는 다양한 종류의 클라이언트를 지원하기 위해서 이종 클라이언트에 대한 스트림 캐시를 위한 방법들이 제안되었다. 가장 간단한 해결 방법으로 각 클라이언트에 맞춰 다른 포맷과 비율로 스트림들을 미리 복제하는 것이 있으나 이것은 저장 공간과 대역폭 요구가 매우 높다. 이것의 대안으로 클라이언트의 요청이 있을 시 그 요구 조건에 따라 인코딩 포맷과 비율로 트랜스코드 하는 방법이 나왔다. 하지만 이 방법은 트랜스코딩을 위해 큰 계산 오버헤드를 가지므로 프록시가 많은 클라이언트를 지원하는 것을 저해한다. 마지막으로 나온 대안은 LC(Layered Coding)을 이용한 계층화된 인코딩과 전송 방법[11][12]과 LC를 응용한 FGS(Fine Grained Streaming) 기반 서비스 방법[13]이 있다. 이러한 LC류는 원본 미디어 객체를 여러 가지 계층으로 압축 하는 것으로 가장 중요한 특징을 가진 '기본계층(Base Layer)'를 기본적으로 받아야하는 제약 조건이 있다. 그러므로 전송 지연이 있는 인터넷 환경에서의 LC류를 이용한 스트리밍 서비스는 부적합하다.

본 논문은 앞서 언급한 문제점들을 극복하기 위해서 MDC(Multiple Description Coding)를 응용한 스트리밍 서비스를 제공한다. MDC는 n개의 디스크립션으로 인코딩한 것으로 서로 내용이 중복되기 때문에 각 디스크립션은 동등한 중요성을 가진다. 즉, 어느 하나를 잃더라도 계속적인 스트리밍 서비스가 가능하기 때문에, MDC를 응용한 것은 전송 지연이 있는 스트리밍 서비스에 적합하다.

또한, 우리는 프록시의 캐싱 공간(Caching space)을 좀 더 효율적으로 사용하고 부하 분산, 데이터 이용 가능성 등의 성능을 높이기 위해 협업 프록시(Cooperative Proxy)를 사용한다. 기존의 단일 프록시를 사용한 스트리밍 서비스는 비디오 데이터의 일부분만을 캐시 하는 것으로 서비스 대기시간(Service Latency)을 줄여주지만 대부분의 데이터를 서버가 여전히 제공하고 있기 때문에 서버 상의 많은 부하를 해결하지 못하는 단점이 있다. 이것을 쉽게 해결하기 위해서는 단일 프록시 내의 캐싱 공간을 상당히 크게 해주는 방법이 있지만 많은 어플리케이션들을 처리하기 위해서 엄청나게 비싼 비용을 지불하게 되기 때문에 이 방법은 비현실적이다. 그러므로

협업 프록시를 사용하여 각 프록시에 비디오 데이터를 분산 저장하여 전체 데이터가 캐싱되어 있는 것처럼 응용하는 것으로 앞의 문제점을 해결할 수 있다. 또한 서버의 고장 시에도 데이터에 대한 높은 이용 가능성을 가지며 저장장치의 부하를 균등하게 할 수 있다.

우리는 MDC를 이용한 협업 프록시를 통하여 효율적인 스트리밍 서비스를 제공한다.

본 논문은 다음과 같이 구성된다: 2장은 기존의 LC를 이용한 스트리밍 서비스와 MDC를 살펴보고, 3장에서는 우리가 제안하는 효율적인 스트리밍 서비스를 하기 위한 MDC를 이용한 협업 프록시 구조를 설계한다. 마지막으로 4장에서는 결론 및 향후 과제를 보여준다.

2. 관련 연구

일반적인 미디어 코더가 하나의 비트 스트림을 생성하는 것과 반대로 LC(계층화 코딩)과 MDC는 하나의 미디어 소스를 두 개 이상의 비트 스트림으로 만들어 피할 수 없는 전송 에러에 강인하게 하는 소스 코딩 방법[24]이다.

2.1 계층화 코딩(LC)을 기반으로 한 스트리밍 서비스

이번 장에서는 LC(Layered Coding)를 기반으로 한 스트리밍 서비스를 소개한다. LC는 스트리밍 서비스를 좀 더 효율적으로 하기 위해서 나온 최근 연구 방안으로써 이종 클라이언트에 대한 스트리밍 서비스를 지원한다. LC는 계층화된 인코딩과 전송 방법을 사용하는 것으로 원본 미디어 콘텐츠를 여러 가지 계층으로 압축한다. 가장 중요한 계층으로 '기본계층(BL : Base Layer)'가 있으며 이것은 콘텐츠의 가장 중요한 특징들을 표현한 데이터를 포함한다. 그리고 추가적인 계층으로 '강화계층(EL : Enhancement Layers)'가 있다. EL은 좀 더 좋은 품질로 재정의 할 수 있는 데이터를 포함하는 것으로 하나씩 추가 할수록 좀 더 좋은 품질의 서비스를 제공 받을 수 있다.

이종 클라이언트들은 환경에 따라 BL과 EL의 수를 요청하며 그에 비례한 서비스 품질을 제공 받는다.

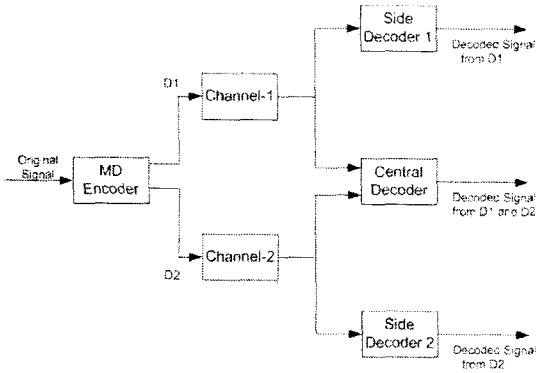
계층화된 캐싱을 위해, Kangasharju 등[11]은 캐시된 부분은 어느 정도 정적이며 오직 완전한 계층들만이 캐시된다고 가정하였다. 전체 성능을 최대화하기 위해, 그들은 어떤 콘텐츠의 어떤 계층을 캐시할 것인지에 대한 결정을 확률적인 방법을 기반으로 하여 개발하였다. 인기 스트림들이 프록시 서버에 캐시 되고 높은 EL 보다는 낮은 BL을 캐시한다. 동적인 환경에서의 계층화된 스트리밍을 위해, Rejaie 등[12]은 이용 가능한 대역폭과 캐시 공간의 효율적인 이용을 위하여 세그먼트 기반 캐시 교체 정책을 연구하였다. 프록시는 각 계층에서 각각의 데이터의 인기도를 유지한다. 캐시된 계층들의 품질이 요청 클라이언트에게 전송 가능한 최대 품질보다 낮을 때, 프록시는 프리패칭 위도우 내에 존재하지 않는 세그먼트들을 서버에게 요청한다. 캐시 교체 정책에서, 교체될 계층은 인기도를 기반으로 하여 선정된다.

이러한 기존 계층화된 스트리밍 구조의 중대한 단점은 계층의 수가 2 또는 3 개의 작은 수로 구성되어 있으므로 이종 클라이언트에 대한 적응 정도가 제한된다는 것이다. 최근 이것을 극복하기 위해 제안된 것으로 LC를 응용한 FGS(Fine Grained Streaming) 기반 서비스[13]가 있다. 이것은 인코딩 비율을 컨트롤하기 때문에 클라이언트의 이용 가능한 대역폭을 거의 완전히 이용하여 적응 정도를 높이지만, 여전히 가장 중요한 BL을 기본적으로 받아야하는 제약이 있다. 만약 BL을 전송하지 못한다면 다른 계층들(enhancement layers)은 소용없게 된다. LC/FGS 기반 서비스는 전송 에러가 거의 없는 환경에서 유리하다. 현재 인터넷 환경은 거의 전송 에러가 없는 환경이지만 스트리밍 서비스에서는 전송 지연이 곧 전송 에러가 되기 때문에 LC/FGS 기반 서비스는 미디어 스트리밍 서비스에서는 부적합하다. 또한 다중 스트리밍의 경우 각 계층을 다른 서버/저장장치에 저장하여 서비스함으로 BL을 저장한 서버/저장장치에 문제가 발생하면 서비스를 전혀 하지 못하며, base layer를 전송하는 서버/저장장치에서 부하가 크기 때문에 부하가 집중 된다.

2.2 다중 디스크립션 코딩(MDC)

MDC(Multiple Description Coding)는 벨 연구소에서 개발된 방법이다[14]. 그 후에 소스 신호에 대한 MDC는 정보 이론 분야와 신호 처리 분야에서 널리 연구되어 왔다[15,16]. 신호 처리 분야에서 MDC의 개념은 음성 압축에 차례로 적용되었으며[17] 그 후에 영상과 동영상 압축에 차례로 적용되었다[18,19]. MDC는 오디오/비디오 신호를 1개 이상의 스트림으로 분할하거나 n개의 디스크립션(description)으로 인코딩하여 각각을 독립적인 경로로 전송하는 메소드를 의미한다. 즉, 하나의 스트림을 여러 개의 스트림으로 분할하고 각 스트림에는 자신의 정보와 다른 스트림에 관한 부가 정보를 추가하여 어느 하나의 스트림을 잃더라도 나머지 스트림으로 복구할 수 있고 모두를 수신하게 되면 송신 시와 똑같은 품질의 컨텐트를 생성할 수 있는 코딩 기술이다[20]. 원본 컨텐트와 MDC로 인코딩된 후의 디스크립션의 질적 관계는 QC가 원본 컨텐트의 품질이고 QD_1~QD_n은 디스크립션의 품질일 경우 다음과 같이 표현될 수 있다: $QC = QD_1 \cup QD_2 \cup \cdots QD_n$. 즉, 컨텐트 C의 품질은 각 디스크립션을 모두 전송 받아 디코딩 한 후의 질과 같다. 그러므로 LC와는 달리 각 비트 스트림은 서로 독립적이며 어느 것이 더 중요한 것이 없이 각각 똑같은 중요도를 가지며 어떤 스트림을 수신하든 재생이 가능하고 수신되는 스트림이 많을수록 재생 품질이 좋아지는 장점이 있다[21].

(그림 1)은 2개의 디스크립션으로 코딩되는 시스템을 나타낸다. MDC 인코더는 2개의 디스크립션을 생성하고 이들을 두 개의 독립적인 채널을 통해서 전송한다. 만일 두 채널 모두에서 에러가 발생하지 않은 경우 중앙 디코더(central decoder)는 손상되지 않은 두 개의 디스크립션을 전송받고 이를 하나의 고품질 신호로 재구성하게 된다. 만일 하나의 채널만이 에러가 발생하지 않은 경우에는 주변 디코더(side decoder)가 하나의 저품질이지만 수용 가능한 신호로 재구성하게 된다[22].

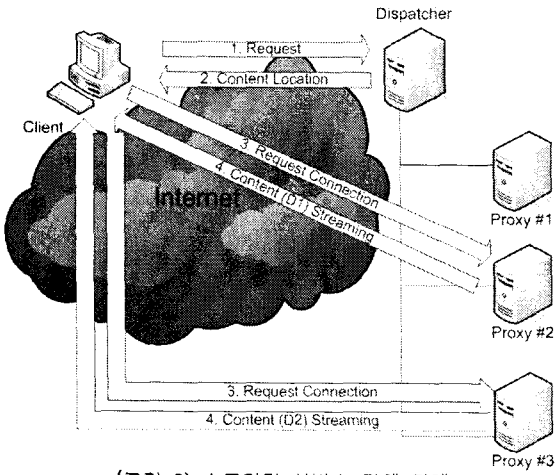


(그림 1) 2개의 디스크립션으로 구성된 MDC 시스템 [22]

3. MDC 컨텐츠 스트리밍을 위한 협업 프록시 구조

우리는 협업 프록시를 기반으로 한 MDC를 이용한 다중 디스크립션 비디오 스트리밍 서비스를 제안한다. 먼저 기본적인 스트리밍 서비스의 단계를 살펴보고 서비스 구현을 위한 메시징 메커니즘(Messaging Mechanism)을 구성을 본다. 추가적으로 캐시 선반입(Cache prefetching)과 교체(replacement) 정책을 제안한다.

3.1 스트리밍 서비스 단계



(그림 2) 스트리밍 서비스 전체 단계

(그림 2)는 MDC를 이용한 스트리밍 서비스를 나타낸 것으로 다음과 같은 단계를 거쳐 제공된다.

1) 사용자는 서비스 받을 컨텐츠를 선택한다.

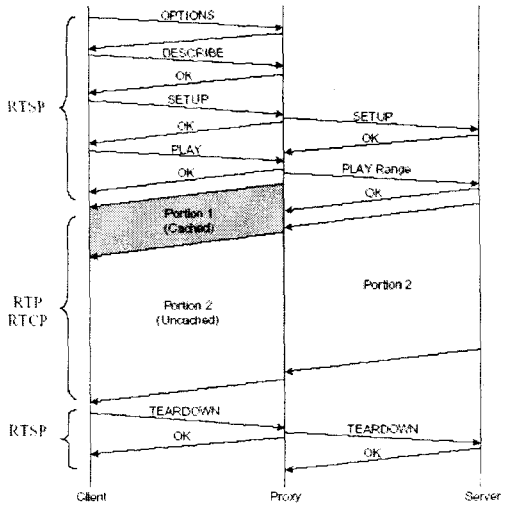
2) 사용자의 요청은 디스패처로 전송된다.

3) 디스패처 서버는 사용자의 요청에 해당하는 디스크립션의 이름과 위치(일반 프록시들)를 사용자에게 전송한다.

4) 사용자는 디스패처로부터 전달받은 일반 프록시들의 정보를 바탕으로 각 디스크립션에 대해 해당 위치로 RTSP 요청을 수행한다.

5) 요청 받은 일반 프록시들은 스트리밍 서비스를 수행한다.

(그림 3)은 스트리밍을 위한 RTSP(Real-Time Streaming Protocol) 연산을 보여준다.



(그림 3) RTSP operation for streaming [2]

3.2 구현 디자인

우리는 이번 장에서 MDC를 이용한 협업 프록시 구조에 대한 디자인을 제안한다. 우리의 제안된 구조는 클러스터형 협업 프록시를 기반으로 하며, 클러스터에 포함된 모든 일반 프록시들을 관리하는 디스패처(Dispatcher)와 그 일반 프록시들 사이의 메시징 메커니즘(Messaging Mechanism)을 통하여 서비스 구현할 수 있다. 제안된 구조는 7가지의 다른 패킷 형식(Packet Type)으로 메시지들을 분류한다: REQ, FOUND, FIND, TRANS, PLAY, SERV, QUIT. 서비스를 제공받기 원하는 클라이언트 C는 자신이 속한 서브넷(Sub-net)의 디스패처에게 REQ 패킷을 보냄으로 원하는 비디오에 대한 서비스를 요청한다. 그 디스패처는 C가 보내온 REQ 패킷을 보고 요청 비디오를 가지고 있는 일반 프록시가 있는지 검사하고 있다면 디스패처는 FOUND 패킷으로 그 프록시의 위치정보를 사용자에게 전달한다. 사용자는 선정된 프록시에게 PLAY 패킷을 보내어 서비스를 요청하고 그 프록시는 SERV 패킷으로 스트리밍 서비스를 시작하게 된다. 만약 디스패처는 REQ 패킷에 적합한 일반 프록시를 찾지 못했을 경우, 디스패처는 데이터를 저장할 적합한 일반 프록시를 선정하고 원본 서버(Source Server)에 FIND 패킷을 보내어 데이터를 찾고 원본 서버는 TRANS 패킷으로 선정된 적당한 일반 프록시에게 데이터를 전송하고 저장한다. 만약 C가 서비스를 더 이상 받지 않길 원한다면 QUIT 패킷을 서비스를 제공하고 있는 일반 프록시에게 전달하여 서비스를 중지한다. 다음은 서비스를 제공하기 위한 패킷 형식과 디스패처와 일반 프록시의 데이터 구조(Data Structure)를 상세히 기술하고 서비스 처리과정을 살펴본다.

3.2.1 패킷 형식과 데이터 구조

(1) 패킷 형식

REQ: 비디오를 요청하기 위해, 사용자는 REP 패킷을 그것의 디스패처에게 보낸다. 각 REQ 패킷은 다음과 같은 정보를 포함한다: ① ClientIP : 사용자의 유일한 IP ② VideoID : 요청 Video의 유일한 ID ③ NumOfDes : 클라이언트 요건에 맞는 Description 수

FOUND: 디스패처는 요청 비디오 데이터를 가진 일반 프록시들의 위치정보를 사용자에게 알려준다. FOUND 패킷은 다음과 같은 정보를 포함한다: ① ProxyIDs : 데이터 가진 적합한 일반 프록시들의 위치 정보 ② DescID : 비디오의 전송할 디스크립션 ID

PLAY: 사용자는 요청 비디오 데이터를 가진 일반 프록시들에게 서비스 제공을 요구한다. 각 PLAY 패킷은 다음과 같은 정보를 포함한다: ① ClientIP : 비디오 요청 사용자의 유일한 IP ② DescID : 비디오의 전송할 디스크립션 ID

SERV: 이것은 비디오 데이터 패킷이다. 요청받은 일반 프록시들은 사용자에게 이 패킷을 전송함으로써 서비스를 수행한다. 각 SERV 패킷은 다음과 같은 헤더 필드들을 포함한다: ① ProxyID : 서비스를 제공하고 있는 일반 프록시의 ID ② ClientIP : 서비스할 사용자의 IP ③ DescID : 서비스할 비디오의 디스크립션 ID

FIND: 일반 프록시에게 요청 비디오 데이터가 없다면 디스패처는 데이터를 저장할 적합한 일반 프록시를 찾고 원본 서버에게 FIND 패킷을 보내어 비디오 데이터를 찾는다. FIND 패킷은 다음과 같은 정보를 포함한다: ① ProxyIDs : 데이터를 저장할 적합한 일반 프록시들의 ID ② VideoID : 요청 비디오의 유일한 ID ③ DescList : 디스패처가 유지하고 있는 일반 프록시들의 디스크립션 리스트

TRANS: 원본 서버는 FIND 패킷을 통해 받은 정보를 통해 적합한 일반 프록시들에게 비디오 데이터를 전송한다. TRANS 패킷을 다음과 같은 헤더 필드들을 포함한다: ① ProxyID : FIND 패킷에 포함된 각 ProxyID ② VideoID: 요청 비디오의 유일한 ID ③ DescID : 비디오의 전송할 디스크립션 ID ④ SegmID : 디스크립션의 순차적인 세그먼트 ID

QUIT: 사용자가 서비스 세션의 참여를 중단하고자 할 경우 서비스를 제공하고 있는 일반 프록시들에게 이 패킷을 보낸다. QUIT 패킷은 다음과 같은 정보를 포함한다: ①ProxyID : 서비스를 제공하고 있는 일반 프록시 ID ② ClientIP, VideoID, DescID : 이 필드들은 중단하길 원하는 서비스를 확인가능

(2) 데이터 구조

디스패처(Dispatcher)는 일반 프록시에 대한 정보를 주기적으로 모니터링 하여 유지한다.

LOAD TABLE (LoadTbl): 일반 프록시의 부하 정보로써 다음과 같은 필드를 포함한다: ① ProxyID : 일반 프록시를 구별하기 위한 ID ② LoadOfProxy : 일반 프록시 부하 ③ LoadOfLink : 일반 프록시와 디스패처 사이의 링크 부하

CACHE DIRECTORY (CacheDir): 일반 프록시가 캐시하고 있는 데이터 정보를 유지하며 다음과 같은 속성을 포함한다: ① ProxyID : 일반 프록시를 구별하기 위한 ID ② VideoID : 비디오를 구별하기 위한 ID ③ DescID : 비디오를 구성하는 디스크립션의 ID ④ SegmID : 디스크립션을 구성하는 세그먼트 ID

SERVING TABLE (ServingTbl): ① ProxyID : 일반 프록시를 구별하기 위한 ID ② ClientIP : 서비스 중인 사용자의 유일한 IP ③ ServDescList : 서비스 중인 디스크립션 리스트

USER TABLE (UserTbl): ① IP : 서비스 중인 사용자 IP ② VideoID : 요청 비디오 ID ③ DescBM : 비디오의 디스크립션 비트맵(서비스 중=1, 아니면=0) ④ CurPos : 현재 서비스 되고 있는 위치 ⑤ PrefCachRate : 프리패칭 윈도우의 캐싱율

일반 프록시는 클러스터에 속한 프록시로써 사용자에게 직접적으로 서비스하기며 인기도에 따른 캐시의 교체 알고리즘을 위해 다음과 같은 정보를 유지한다.

SERVICE TABLE (ServiceTbl): 일반 프록시가 유지하고 있는 비디오에 대한 정보와 인기도를 유지한다: ① VideoID : 비디오를 구별하기 위한 ID ② DescID : 비디오를 구성하는 디스크립션의 ID ③ SegmID : 디스크립션을 구성하는 세그먼트 ID ④ Status : 세그먼트의 인기도

3.2.2 서비스 처리과정

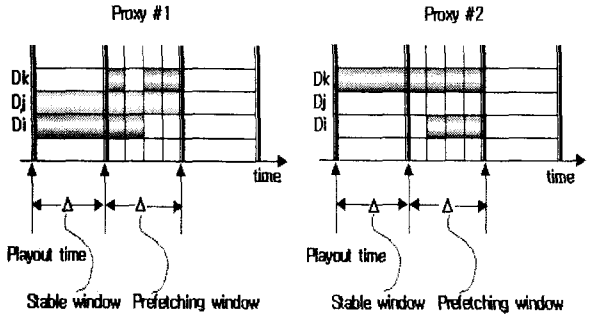
서비스를 제공받길 원하는 사용자는 자신의 디스패처에게 REQ 패킷을 보냄으로 원하는 비디오에 대한 서비스를 요청한다. 그 디스패처는 자신의 캐시 디렉토리 *CacheDir*을 검사하여 사용자가 원하는 디스크립션 수 이상으로 데이터를 포함하는 일반 프록시가 있다면, 각 프록시의 부하에 대한 정보를 가진 *LoadTbl*을 보고 요청 디스크립션 수만큼의 부하가 가장 적은 일반 프록시들을 선정하고 *ServingTbl*과 *UserTbl*에 사용자 정보를 추가하고, 디스패처는 선정된 프록시들의 위치 정보를 포함한 FOUND 패킷을 사용자에게 전달하게 된다. 사용자는 그 일반 프록시들에게 PLAY 패킷을 보냄으로써 서비스를 요구한다. PLAY 패킷을 받은 선정된 프록시들은 SERV 패킷을 통해 직접 사용자에게 스트리밍 서비스를 시작하고 디스패처의 *ServingTbl*과 *UserTbl*의 정보를 갱신 시킨다.

만약 디스패처의 *CacehDir*에 비디오 데이터를 가진 프록시의 수가 하나도 없거나 REQ 패킷의 요청 디스크립션 수인 NumOfDes 보다 작다면, 부족한 수만큼을 일반 프록시에게 캐시하기 위해 *LoadTbl*을 보고 부하가 적은 적당한 프록시들을 선택한다. 그 후 디스패처는 비디오 데이터를 저장하고 있는 원본 서버에게 FIND 패킷을 보내고 원본 서버는 선정된 프록시들에게 TRANS 패킷을 보내어 데이터를 전송하고 저장하게 된다. 디스패처는 일반 프록시들을 주기적인 모니터링 함으로써 *LoadTbl*과 *CacheDir*를 갱신하게 되며, 일반 프록시는 사용자의 요청에 의해 서비스하는 비디오 데이터들의 인기도(Status)를 측정하게 된다.

3.3 캐시 선반입 정책 (Cache Prefetching Policy)

웹 프록시의 경우 사용자의 요청에 대한 데이터가 프록시에 없다면 원본 서버로부터 데이터 전체를 전송 받는다. 전체 데이터를 캐시하는 것은 이미지나 텍스트 같은 작은 사이즈의 데이터이기에 가능하다. 하지만 우리가 주장하는 스트리밍 프록시에서 비디오 데이터를 캐시하기 위해서는 그 사이즈가 매우 방대하기에 전체 데이터를 캐시하는 것은 캐시 공간 및 대역폭을 낭비할 가능성이 매우 높다. 그리고 모든 사용자가 처음부터 끝까지 보지 않을 수도 있으며 동시성을 높일 수 없다.

그러므로 스트리밍 프록시의 캐시 공간을 좀 더 효율적으로 사용하기 위해서 일반적으로 세그먼트 단위로 캐시한다. 사용자가 현재 보고 있는 세그먼트이고 프록시에 없다면 원본 서버로부터 가져오게 하는 방법으로 현재 세그먼트에 대한 유무를 재생 순간에 판단하면 전송 지연이 발생하게 된다. 따라서 사용자의 요청에 따라 미리 가져올 세그먼트를 판단하여 원본 서버로부터 미리 가져오는 선반입을 수행한다. 세그먼트를 너무 미리 가져오면 전체 데이터 단위로 처리하게 되며 너무 늦게 가져오면 순수한 세그먼트 단위로 가져오게 되는 문제점을 가진다. 그렇기 때문에 우리는 프리패칭 윈도우를 사용하여 세그먼트를 적당한 간격으로 미리 가져오게 한다.



(그림 4) Prefetching window

본 논문에서 선반입 정책은 다음과 같이 처리된다. 디스패처는 사용자에게 계속적인 서비스를 제공할 프록시를 선택하기 위해, *UserTbl*의 서비스되고 있는 비디오의 각 디스크립션 별 PrefCachRate를 확인한다. 첫 번째 안정된 윈도우(Stable window)에서 재생을 시작할 때 두 번째 윈도우에서 선반입(Prefetching window)을 시작한다. 어느 프록시에서 어떤 디스크립션을 계속적으로 서비스할지 선택해야만 선반입을 할 수 있기 때문에, 프리패칭 윈도우 내에 포함된 데이터의 저장된 양(PrefCachRate)을 보고 높은 값을 가진 프록시의 디스크립션을 (원본서버/클러스터 내에서) 선반입한다.

(표 1) 각 디스크립션의 PrefCachRate

	Proxy #1		Proxy #2	
Desc#	Stable window	Prefetching window	Stable window	Prefetching window
Di	100	50	0	75
Dj	100	100	0	0
Dk	0	75	100	100

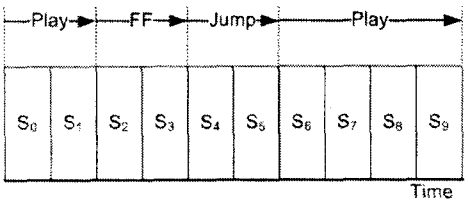
비디오 V에 대한 3개의 디스크립션이 (그림 4)와 같이 P1(Proxy #1)과 P2(Proxy #2)에 캐시되어 있다. 사용자가 3개의 디스크립션을 디스패처에게 요청할 경우, 디스패처는 *UserTbl*에서 각 디스크립션의 PrefCachRate 값을 확인한다. 각 프록시의 PrefCachRate가 (표 1)과 같을 때, 디스패처는 Stable window인 P1의 Di와 Dj와 P2의 Dk를 재생하고 P2는 P1의 Di로부터 첫 번째 세그먼트를 선반입한다.

3.4 캐시 교체 정책 (Cache Replacement policy)

기존의 프록시를 이용한 스트리밍 서비스와 관련된 연구들은 프록시의 저장 공간이 부족할 때 새로운 스트림을 캐시하기 위하여 LFU(Least-Frequently-Used), FIFO(First-in, First-out) 등의 교체 정책을 사용하였다 [11][23]. 본 논문은 Prefix caching, Sliding caching 등의 기존 연구들과는 달리 MDC를 사용하여 캐시 공간을 효율적으로 사용하고 있다. MDC는 앞서 언급한 것과 같이 n개의 디스크립션으로 구성되며 각 디스크립션은 여러 개의 세그먼트로 구성된다. 우리는 세그먼트의 인기도를 기반으로 교체 정책을 수행한다.

일반 프록시는 저장하고 있는 각 디스크립션의 세그먼트에 대한 인기도를 측정하여 *ServiceTbl*의 Status에 저장한다. 새로운 스트림이 저장되길 원하고 저장 공간이 꽉 찼을 때 프록시는 디스패처의 *ServingTbl*을 확인하여 어떤 사용자에게도 서비스하고 있지 않는 디스크립션(ClientCount=0)을 확인하여 그 디스크립션의 인기도(Status)가 가장 낮은 세그먼트를 선택하여 교체한다. 만약 가장 낮은 인기도를 가진 세그먼트가 여러 개 있다면 스트림의 끝부분 즉, SegmID가 가장 큰 것을 교체한다.

우리는 일반 프록시에 저장된 각 세그먼트의 인기도를 측정하기 위하여 재생(Play), 빨리 감기(Fast-Forward) 등의 사용자의 VCR 기능에 따른 요청(hit)에 따라 다른 점수를 부가한다.



각 세그먼트(Sk)의 인기도는 다음과 같이 측정하여 일반 프록시의 ServiceTbl의 Status에 저장한다.
(Di: 디스크립션 번호, Sk: 세그먼트 번호, t: 현재 요청 시간, t-1: 이전 요청 시간, user: 요청 사용자 수)

$$hit(Di,Sk)_t = hit(Di,Sk)_{t-1} + user * \begin{cases} 0\ (Jump, no\ operation) \\ 0.5\ (FF, FR) \\ 1\ (Play) \end{cases}$$

4. 결론 및 향후 연구과제

우리는 다중 디스크립션 비디오 스트리밍을 위한 협업 프록시 구조를 제안하였다. 본 논문은 MDC(Multiple Description Coding) 기법을 기반으로 콘텐츠의 디스크립션들을 클러스터된 일반 프록시들에게 균등하게 분산시켜 모든 프록시의 부하가 같아지고, 특정 서버나 프록시의 고장 시에도 데이터에 대한 높은 이용 가능성을 가지므로 끊어지지 않는 스트리밍 서비스를 가능하게 하며, 기존의 단일 프록시를 사용한 스트리밍 서비스가 해결하지 못한 서버 상의 많은 부하를 해결하였다. 또한 일반 프록시들의 정보를 유지하고 있는 디스패처가 요청 사용자의 환경에 적응적으로 스트리밍 서비스를 가능하게 하며, 제안된 구조에 적합한 캐시 선반입 정책과 교체 정책을 제시하여 캐시 공간의 효율성을 높였다.

우리는 보다 더 나은 서비스를 제공하기 위해 클러스터 간의 통신과 동기화 문제, 사용자에게 서비스 되고 있는 디스크립션 수의 편차를 작게 하는 방법과 디스크립션의 조합을 서비스 되고 있는 동안 계속적으로 유지하는 방법 등을 고안 중이다. 또한 제안된 협업 프록시 구조의 성능을 평가하기 위한 실험 단계에 있다.

참고문헌

[1] J. G. Apostolopoulos and S. J. Wee, "Unbalanced multiple description video communication using path diversity", In IEEE International Conference on Image Processing, Oct. 2001.

[2] J. Liu and J. Xu, "A Survey of Streaming Media Caching", available at: www.comp.hkbu.edu.hk/~xujl/streamCaching.pdf, Technical Report, Dec. 2003.

[3] A. Dan, Y. Heights, and D. Sitaram, "A generalized interval caching policy for mixed interactive and long video workloads", In Proc. of SPIE/ACM Conf. on Multimedia Computing and Networking (MMCN'96), San Jose, CA, Jan. 1996.

[4] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams:, In Proc. IEEE INFOCOM'99, New York, NY, Mar. 1999.

[5] S. Jin, A. Bestavros, and A. Iyenger, "Accelerating Internet streaming media delivery using network-aware partial caching", In Proc. IEEE ICDCS'02, Vienna, Austria, July 2002

[6] K. L. Wu, P. S. Yu, and J. L. Wolf, "Segment-based proxy caching of multimedia streams", In Proc. WWW10, HongKong, May 2001.

[7] S. Chen, B. Shen, S. Wee, and X. Zhang, "Adaptive and lazy segmentation based proxy caching for streaming media delivery", Proc. NOSSDAV'03, Monterey, CA, June 2003.

[8] Z. Miao and A. Ortega, "Scalable proxy caching of video under storage constraints", IEEE J. Select. Areas in Comm., Vol.20, no.7, pp.1315-1327, Sep.2002.

[9] H. Fahmi, M. Latif, S. Sedigh-Ali, A. Ghafoor, P. Liu, and I. Hsu, "Proxy servers for scalable interative video support", IEEE Computer, 43(9): 54-60, Sep. 2001.

[10] Z. L. Zhang, Y. Wang, D. Du, and D. Su, "Video staging : A proxy-server-based approach to end-to-end video delivery over wide-area networks", IEEE/ACM Trans. Networking, 8(4) : 429-442, 2000.

[11] J. Kangashariju, F. Hartanto, m. Reisslein, and k. W. Ross, "Distributing layered encoded video through caches", IEEE Trans. Computers, 51(6), pp.622-636, June 2002.

[12] R. Rejaie, H. Yu, M. Handley, and D. Estrin, "Multimedia proxy caching mechanism for quality adaptive steaming applications in the Internet", In Proc. IEEE INFOCOM'00, Tel Aviv, Israel, Mar. 2000.

[13] J. Liu, X. Chu, and j. Xu, "Proxy Cache Management for Fine-Grained Scalable Video Streaming", Proc. IEEE INFOCOM'04, HongKong, Mar. 2004.

[14] Michael Zink, Andreas Mauthe, "P2P Streaming using Multiple Description Coded Video", Proceedings of EuroMicro2002, Sep. 2004.

[15] L. Ozarow, "On a source coding problem with two channels and three receivers", Bell Syst. Tech. J., Vol.59, pp.1909-1921, Dec., 1980.

[16] V. A. Vaishampayan, "Design of multiple description scalar quantizer", IEEE Trans. Inform. Theory, Vol.39, pp.821-834, May. 1993.

[17] A. Ingle and V. A. Vaishampayan, "DPCM system design for diversity systems with applications to packetized speech", IEEE Trans. Speech and Audio Processing, Vol.3, pp.48-57, Jan. 1995.

[18] M. Orchard, Y. Wang, V. A. Vaishampayan, and A. Reibman, "Redundancy rate distortion analysis of multiple description image coding using pairwise correlating transforms", Proc. Int. Conf. Image Processing, pp.608-611, Oct. 1997.

[19] Y. Wang, M. Orchard, V. A. Vaishampayan, and A. Reibman, "Multiple description coding using pairwise correlating transforms", IEEE Trans. Image Processing, Vol.10, pp.351-366, Mar. 2001.

[20] Apostolopoulos, J. G., "Error-resilient Video Compression via Multiple State Streams", In Processings of VLBV'99. Oct. 1999.

[21] 김선호, 송병호, "멀티미디어 콘텐츠의 QoS를 개선한 전송 메커니즘", 정보처리학회 논문지 B, 제12-B권 제2호, 2004. 4.

[22] Yen-Chi Lee, Joohee Kim, Yucel Altunbasak, Russell M. Mercereau, "Layered coded vs. multiple description coded video over error-prone networks", EURASIP signal processing : image communication, Vol.18, pp.337-356, May. 2003.

[23] Duc A. Tran, Kien A. Hua, and Simon Sheu, "A New Caching Architecture for Efficient Video-on-Demand Services on the Internet", IEEE Computer Society, Washington. DC. USA, 2003.