다중 디스크립션 비디오 스트리밍을 위한
협업 프록시 구조 설계

이승은, 한종욱, 한종열, 박유현, 김명석
부산대학교 컴퓨터공학과, 한국전자통신연구원, 부산대학교 정보컴퓨터공학부
(seel@, jwan@, hyan@, glmg@, bakyh@asadal.pusan.ac.kr, etrei.re.kr)

Cooperative Proxy Architectural Design for Multiple Description Video Streaming

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

요 약
최근 인터넷 접속 환경의 발달과 시스템 및 네트워크 성능의 혁신으로 많은 사용자들은 볼수록 이슈가
많은 단순한 정보 요구뿐만 아니라 큰 용량과 많은 대역
폭 사용 등의 특징을 가진 VoD(Video-on-Demand : 주문형 비디오) 서비스를 요구하고 있다. 사용자들은 서버
요청에 대한 훨씬 응답속도 서비스를 발기로 원한다.
이러한 요구는 사용 시간과 서버/네트워크 부하를 줄이기
위한 효율적인 방안으로 클라이언트에서 가까이에 있는 프
록시들에 반변수 사용되는 데이터를 캐시(cache)하는 것
이 있다.
기존 프록시들은 웹 페이지들을 캐시하기 위해 효과적
하지만, 일반적으로 비디오 파일은 웹 페이지보다
수백만 배 더 크기 때문에 하나의 프록시에 캐시되는 비디오 파일은 매우 제한적이며 간단한 서버와 간
활용을 모두 소비하게 된다. 그럴기 때문에 비디오
오디오 파일과 같은 스트리밍 서비스를 위한 자바로
프록시에 대한 연구들이 진행되어 왔다.
 초기의 알티미디어 프록시 연구는 주로 디스크의 값
줄(비디오 파일, 장치 구성 등의 클라이언트)을 대
상으로 서비스 하였다. 값 줄 종류의 클라이언트에 대해
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. 최근
에는 다양한 종류의 클라이언트를 지원하기 위해 초기
클라이언트의 캐시를 위한 방법들이 제안
되었다. 가장 간단한 해결 방법으로 각 클라이언트에 맞
맞춤 프로그램 비율로 스트림들은 미리 복제하는 것이
있으나 이것은 저장 공간과 대역폭 요구가 매우 높다.
이것의 대안으로 클라이언트의 요청이 있을 시 그 요구
조건에 따라 캐시 포맷과 비율로 스트리밍방식은 방
법이 많아진다. 하지만 이 방법은 인터넷등을 이용한
한계 오버헤드를 줄이기 위해 프록시가 많은 클라이언트를
지원하는 것을 저해한다. 이제까지는 매크로 분
LCR(Layered Coding)를 이용한 계층화된 인코딩과 전송
방법[11][12]과 LCB를 이용한 FGS(Fine Grained
Streaming) 기반 서비스 방법[13]이 있다. 이러한 LC
층은 원복 하위 계층을 여러 계층으로 압축하는 것
으로 간출한 구조를 가진 '기본계층(Base Layer)'를
기본으로 하여 압축하는 계층 조각이 있다. 그러므로
전송 지원이 있는 인터넷 환경에서의 LCB를 이용한 스
트리밍 서비스는 적합하다.
본 논문은 앞서 언급한 문제점을 극복하기 위해서
MDC(Multiple Description Coding)를 이용한 스트리밍
서비스를 제공한다. MDC는 낮은 디스크립션으로 인코
딩된 것으로서 모든 정보가 증가되기 때문에 각 디스크립
션은 동등한 중요성을 가진다. 즉, 하나는 둘다로
도 계층한 스트리밍 서비스가 가능하기 때문에, MDC
를 이용한 것은 전송 지원이 있는 스트리밍 서비스에 적
합하다.
또한, 우리는 프록시의 캐시 공간(Caching space)을
 좀 더 효율적으로 사용하고 부하 분산, 데이터 사용 가
능성 등의 성능을 높이기 위해 협업 프록시(Cooperative Proxy)를 사용한다. 기존의 단일 프록시를 사용하는 스트리
밍 서비스는 비디오 데이터의 일부만을 캐시 하는 것으로 서비스 대기 시간(Service Latency)을 줄이자 다만 일부만
다음이의 데이터를 캐시가 여전히 제공하고 있기 때문에
서비스 상의 많은 부하를 해결하지 못하는 단점이 있다.
이것을 해결하기 위해서는 단일 프록시 내의 캐시
공간을 상당히 크게 해주면 방법이 있지만 많은 어려움
케이스들이 급격히 해결하기 위해서 타당하게 비용을 지불
하게 되기 때문에 이 방법은 비현실적이다. 그러므로
협업 프로토콜을 사용하여 각 프로토콜에 비디오 데이터를 분산 저장하여 전체 데이터가 캐시되어 있는 것처럼 속용하는 것으로 앞의 문제점을 해결할 수 있다. 또한 서버의 고정 시스템의 데이터에 대한 높은 사용 가용성을 가지며 저장장치의 비용을 극대화할 수 있다.

우리는 MDC를 이용한 협업 프로토콜을 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 프로토콜 구조를 설계한다. 이지만으로 4장에서는 결론 및 향후 과제를 보여준다.

2. 관련 연구

일반적인 미디어 코딩이 하나의 비트 스트림을 생성하는 것은 반대로 LC(계층화 코딩)과 MDC는 하나의 미디어 소스를 두 개 이상의 비트 스트림으로 만들어 피할 수 없는 전송 에러에 강한 고용량 스오 코딩 방법[24]이다.

2.1 계층화 코딩(LC)을 기반한 한 스트리밍 서비스

이번 장에서는 LC(Layered Coding)를 기반한 한 스트리밍 서비스를 소개한다. LC는 스트리밍 서비스를 좀 더 효율적으로 하기 위해서 새로운 연구 방법으로서 이중 클라우드에 대한 스트리밍 서비스를 제공한다. LC는 계층화된 인코딩과 전송 방법을 사용하는 것으로 설정 미디어 콘텐츠의 여러 가지 계층을 얻는다. 가장 중요한 계층으로 '기본계층(BL; Base Layer)이 가며 이것은 콘텐츠의 가장 중요한 특성을 표현하는 데 이어 흐름한 계층으로, '강화계층 (EL; Enhancement Layers)이 가며, EL은 좀 더 좋은 품질로 재생할 수 있는 데이터를 포함하는 것으로 하나의 추가가 되면 조금 더 좋은 품질의 서비스를 제공 받는다.

이중 클라우드는 한 환경의 BL과 EL을 수송하여 그에 비해 서비스 품질을 제공 받는다.


이러한 기존 계층화된 스트리밍 구조의 한 단점을 계층의 수가 2 또는 3 개의 작은 수로 구성되어 있으므로 이중 클라우드에 대한 적용 정도가 제한되는 것이다. 최근 이것을 극복하기 위해 제안된 것으로 LC를 이용한 FGS(Fine Grained Streaming) 기반 서비스[13]가 있다. 이것은 인코딩 비율을 변동하기 때문에 클라우드의 이용 가능한 대역폭을 계층화로 연과 하여 이용함으로써 적용 정도를 높이고, 타고자 기준으로 변동 가능한 EL을 기본적으로 받아들이는 제한이 없다. 만약 BL을 전송하지 못한다면 다른 계층들이(Enhancement Layers)는 소용없기 때문에, LC/FGS 기반 서비스는 미디어 스트리밍 서비스는 불가능하다. 또한 다른 스트리밍의 경우 계층을 다른 서버/저장장치에 저장하여 서비스하므로 EL을 저장한 서버/저장장치에 문제가 발생하여 서비스를 전혀 하지 못하므로, 대신 EL을 전송하는 서버/저장장치에 서버가 크게 때문에 불가능하게 된다.

2.2 다중 디스크립션 코딩(MDC)


그림 1은 2개의 미디어 코딩을 모든 미디어에 대해 전송하는 단지. MDC는 2개의 미디어 코딩을 생성하고 이를 두 개의 별도로 전송하고 전송할 수 있는 경우 중앙 디스크립션 센터(central decoder)는 손상되지 않은 별도의 디스크립션을 전송받아 이를 하나의 별도로 전송할 수 있게 된다. 모두 하나의 채널만에 대해 전송할 수 있는 경우에는 주변 디스크립션 사이에 하나의 채널만이 제공될 수 있는 것으로 볼 수 있다. 제공된 서비스의 개수를 전송할 수 있는 수에 따라 제공할 수 있는 수로 크게 줄어들게 된다[17].
3. MDC 컨텐츠 스트리밍을 위한 협업 프로토콜 구조

우리는 협업 프로토콜을 기반으로 한 MDC를 이용한 다중 디스크립션 비디오 스트리밍 서비스를 제안한다. 여기서 기본적인 스트리밍 서비스의 단계를 살펴보고 서비스 구현을 위한 메시징 메커니즘(Messaging Mechanism)을 구성할 것이다. 추가적으로 캐시 채색인(Cache prefetching)과 교체(replacement) 정책을 제안한다.

3.1 스트리밍 서비스 단계

(그림 2) 스트리밍 서비스 전체 단계

(그림 2)는 MDC를 이용한 스트리밍 서비스의 단계를 나타낸 것이다. 다음과 같은 단계를 거쳐 제공된다.
1) 사용자는 서비스란을 클라이언트로 선택한다.
2) 사용자의 요청은 디스패처로 전송된다.
3) 디스패처 서버는 사용자의 요청에 해당하는 디스크립션의 이름과 위치(일반 프로토콜)를 사용자에게 전송한다.
4) 사용자는 디스패처로부터 전달받은 일반 프로토콜의 정보를 바탕으로 각 디스크립션에 대해 해당 위치로 RTSP 요청을 수신한다.
5) 요청 받은 일반 프로토콜은 스트리밍 서비스를 수행한다.

3.2 구성 디자인

이번 장에서 MDC를 이용한 협업 프로토콜 구조에 대한 디자인을 제안한다. 우리의 제안된 구조는 클러스터형 협업 프로토콜을 기반으로 하며, 클러스터에 포함된 모든 일반 프로토콜에 관리는 디스패처(Dispatcher)와 그 일반 프로토콜 사이의 메시징 메커니즘(Messaging Mechanism)을 통하여 서비스 구현할 수 있다. 제안된 구조는 7가지의 단계에 나뉘고, Packet Type으로 메시지를 분류한다: REQ, FOUND, FIND, TRANS, PLAY, SERV, QUIT. 서비스를 제공하기 원하는 클라이언트 C는 자신이 속한 서브넷(Sub-net)의 디스패처에게 REQ 패킷을 보내고 싶은 비디오에 대한 서비스를 요청한다. 그 디스패처는 C가 보내는 REQ 패킷을 보고 요청의 비디오를 가지고 있는 일반 프로토콜이 있는지 검사하고 있다면 디스패처는 FOUND 패킷으로 그 프로토콜의 위치정보를 사용제에 전달한다. 사용자는 QUIT패킷을 보내서 서비스를 요청하고 그 프로토콜의 SERV 패킷으로 디스패처 서비스를 시작하게 된다. 만약 디스패처는 REQ 패킷을 적절한 일반 프로토콜 찾지 못했을 경우, 디스패처는 데이터를 저장할 적절한 일반 프로토콜을 선택하고 원본 서버(Source Server)에 FIND 패킷을 보내서 데이터를 찾을 원본 서버는 TRANS 패킷으로 선택된 적절한 일반 프로토콜에게 데이터를 전송하고 저장한다. 만약 C가 서버로 더 이상 받지 않으면서 QUIT 패킷을 서비스를 제공하고 있는 일반 프로토콜에 전달하여 서비스를 중단한다. 다음은 서비스를 제공하기 위한 패킷 형식과 디스패처의 일반 프로토콜의 메시지 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 그들의 디스플레이에 보낸다. 각 REQ 패킷은 다음과 같은 정보를 포함한다: ① ClientIP: 사용자의 유일한 IP ② VideoID: 요청 Video의 유일한 ID ③ NumOfDes: 클라이언트 요청에 맞는 Description 수

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

PLAY: 사용자는 요청 비디오 데이터를 가져 일반 프로토시들에게 서비스 제공을 요구한다. 각 PLAY 패킷은 다음과 같은 정보를 포함한다: ① ClientIP: 비디오 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 메타 헤더들을 포함한다: ① ProxyID: 서비스를 제공하고 있는 일반 프록시의 ID ② ClientIP: 서비스 사용자의 IP ③ DescID: 서비스할 비디오의 디스크립션 ID

FIND: 일반 프로토시에 요청 비디오 데이터가 없다면 디스플레이는 데이터를 저장할 적합한 일반 프로토시를 찾고 원본 서버에게 FIND 패킷을 보내 비디오 데이터를 찾는다. FIND 패킷은 다음과 같은 정보를 포함한다: ① ProxyIDs: 데이터를 저장할 적합한 일반 프로토시들의 ID ② VideoID: 요청 비디오의 유일한 ID ③ DescList: 디스플레이가 유지하고 있는 일반 프로토시들의 디스크립션 리스트

TRANS: 원본 서버는 FIND 패킷을 통해 받은 정보를 통해 적합한 일반 프로토시들에게 비디오 데이터를 전송한다. TRANS 패킷은 다음과 같은 메타 헤더들을 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 서버를 제공하고 있는 일반 프로토시들에게 이 패킷을 보내는다. QUIT 패킷은 다음과 같은 정보를 포함한다: ① ProxyID: 서비스를 제공하고 있는 일반 프로토시 ID ② ClientIP, VideoID, DescID: 이 필드들은 중단할 원하는 서비스를 확인 가능

(2) 데이터 구조

디스플레이(Dispatcher)는 일반 프로토시에 대한 정보를 주기적으로 모니터링하여 제공하는。

LOAD TABLE (LoadTbl): 일반 프로토시의 부가 정보로 사용자에게 전달하기 위해 다음과 같은 정보를 포함한다: ① ProxyID: 일반 프로토시의 수치 ID ② LoadOfProxy: 일반 프로토시의 부가 정보 정보 ③ LoadOfLink: 일반 프로토시의 부가 정보 정보

CACHE DIRECTORY (CacheDir): 일반 프로토시가 캐시 하지 않은 데이터 정보를 유지하며 다음과 같은 속성을 포함한다: ① ProxyID: 일반 프로토시를 구별하기 위한 ID ② VideoID: 비디오를 구별하기 위한 ID ③ DescID: 비디오를 구별하는 디스크립션의 ID ④ SegmID: 디스크립션을 구별하는 세그먼트 ID

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

USER TABLE (UserTbl): ① IP: 사용자의 유일한 IP ② VideoID: 요청 비디오의 ID ③ ServDesc: 비디오의 디스크립션 비트맵(서비스 중=1, 아니면=0) ④ CurPos: 현재 서비스 되고 있는 위치 ⑤ PrefCachRate: 프리페치 로드 유도의 캐시율

일반 프로토시는 클러스터에 속한 프로토시로서 사용자에게 적절적으로 서비스하기 위해 인기도에 따른 서비스의 제공을 결정하기 위해 다음과 같은 정보를 유지한다: ① VideoID: 비디오를 구별하기 위한 ID ② DescID: 비디오를 구별하는 디스크립션의 ID ③ SegmID: 디스크립션을 구별하는 세그먼트 ID ④ Status: 세그먼트의 인기도

3.2.2 서비스 처리과정

서비스를 제공받기 원하는 사용자는 자신의 디스플레이에 요청 패킷을 보낸다. 요청 패킷을 보낸 비디오에 대한 서비스를 제공한다. 그 디스플레이는 자신의캐시 디렉토리(CacheDir)를 검색하여 사용자에게 보이는 디스크립션을 찾는다. 일반 프로토시의 수 이상으로 데이터를 포함하는 일반 프로토시가 있다면, 각 프로토시의 부가 정보를 가진 LoadTbl을 보고 요청 디스크립션 수만큼의 부가 정보가 가장 적은 일반 프로토시를 선정하고 및 ServingTbl에 사용자 정보를 추가하고, 디스플레이는 선정된 프로토시들의 위치정보를 포함한 FOUND 패킷을 사용자에게 전달한다. 사용자는 그 일반 프로토시들에게 PLAY 패킷을 보내서 서비스를 요구한다. PLAY 패킷은 선정된 프로토시들은 SERV 패킷을 통해 적절한 사용자에게 스토리밍 서비스를 제공하고 디스플레이의 ServingTbl에 UserTable의 정보를 갱신한다. 만약 디스플레이의 CacheDir에 비디오 데이터가 있는 프로토시의 수가 하나 남지 않거나 REQ 패킷의 요청 디스크립션 ID가 다른 user를 보고 부가 정보를 가진 적절한 프로토시들을 선택한다. 그 후 디스플레이는 비디오 데이터를 저장하고 있는 원본 서버에게 FIND 패킷을 보내고 원본 서버는 선정된 프로토시들에게 TRANS 패킷을 보내 데이터를 전송하고 저장하게 된다.
3.3 캐시 선반임 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 데이터에가 가능하다. 하지만 뉴리가 주장하는 스토리밍 프록시에서 비디오 데이터를 캐시하기 위해서는 그 사이즈가 매우 대단지기에 전체 데이터를 캐시하는 것은 캐시 공간 및 대역폭을 낭비할 가능성이 매우 높다. 그리고 모든 사용자가 처음부터 끝까지 보지 않을 수도 있으며 동시에쓰는 빈도가 낮을 수 있다.

그러므로 스토리밍 프록시의 캐시 공간을 좀 더 효율적용 수용하기 위해서 일반적으로 세그먼트 단위로 캐시한다. 사용자가 현재 보고 있는 세그먼트보다 프록시에 없다면 원본 서버로부터 가져오기 하는 방법으로 현재 세그먼트에 대해 유후를 재생할 수 있도록 판단해 전송여부를 결정한다. 따라서 사용자의 요청에 따라 여러 가지 세그먼트를 판단하여 원본 서버로부터 여러 가지로는 선반임을 수행한다. 세그먼트를 너무 자세히 자세히 관리하면 전체 데이터 단위로 처리하게 되어 너무 쉽게 가져오거나 선반한 세그먼트 단위로 가져오게 되는 문제가 생긴다. 그렇기 때문에 우리는 프리페치 원도우를 사용하여 세그먼트를 적당한 간격으로 미리 가져오게 한다.

비디오 V에 대한 3개의 디스크립션 (그림 4)와 같이 P1(Proxy #1)과 P2(Proxy #2)에 캐시되어 있다. 사용자가 3개의 디스크립션을 디스크페치에 요청할 경우, 디스크페치는 UserTb에서 각 디스크립션의 PrefCachRate 값을 확인한다. 각 프록시의 PrefCachRate가 (표 1)과 같은 경우, 디스크페치는 Stable window인 P1의 DI와 P2의 Dk를 재생하고 P2는 P1의 DI로부터 첫 번째 세그먼트를 선반임한다.

3.4 캐시 교체 정책 (Cache Replacement policy)

기존의 프록시를 이용한 스토리밍 서비스와 관련된 연구들은 프록시의 저장 공간이 부족할 때 새로운 스토리밍을 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 인기도 기준으로 교체 정책을 수행한다.

일반 프록시는 저장하고 있는 각 디스크립션의 세그먼트에 대한 인기도를 측정하여 ServiceTb의 Status에 저장한다. 새로운 스토리밍이 저장되길 원하고 저장 공간이 갈 때 포크리는 이 대체의 ServiceTb를 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 가지 세그먼트가 여러 개 있다면 스토리밍의 일부분에 SegmID가 가장 큰 것을 교체한다.

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

![그림 4 Prefetching window](image)

본 논문에서는 선반임 정책은 다음과 같이 처리한다. 디스크페치는 사용자에게 계속적인 서비스를 제공할 프록시를 선전하기 위해, UserTb의 서비스되고 있는 비디오의 각 디스크립션 별 PrefCachRate를 확인한다. 첫 번째 선전된 원도우(Stable window)에서 재생할 시작할 때 두 번째 원도우에서 선반임(Prefetching window)을 시작한다. 어느 프록시에서 여러 디스크립션을 계속적으로 서비스할 수 있어야 선반임을 할 수 있기 때문에, 프리페치 원도우 내에 포함된 데이터의 저장된 양 (PrefCachRate)을 보고 높은 값을 가진 프록시의 디스크립션을 (원본서버/클러스터 내에서) 선반임한다.

![표 1 각 디스크립션의 PrefCachRate](image)

<table>
<thead>
<tr>
<th>Proxy #1</th>
<th>Stable window</th>
<th>Prefetching window</th>
<th>Proxy #2</th>
<th>Stable window</th>
<th>Prefetching window</th>
</tr>
</thead>
<tbody>
<tr>
<td>DI</td>
<td>100</td>
<td>50</td>
<td>100</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>Dk</td>
<td>100</td>
<td>100</td>
<td>75</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

각 세그먼트(Sk)의 인기도는 다음과 같이 측정하여 일반 프록시의 ServiceTb의 Status에 저장한다.

(Di: 디스크립션 번호, Sk: 세그먼트 번호, t: 현재 요청 시간, t−1: 이전 요청 시간, user: 요청 사용자 수)

\[
\text{hit}(Di, Sk) = \text{hit}(Di, Sk) + \text{user} \times 0.5 \quad (FF, FR) \\
(1 \text{ Play})
\]
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 클러스터링한 일반 프록시들에게 균등하게 분산 시키며 모든 프록시의 부하가 같아지고, 특정 서비스가 프록시의 사용부하에 따른 높은 이용 가능성을 가지 못하게 하는 스트리밍 서비스를 가능하게 하며, 기존의 단일 프록시를 사용한 스트리밍 서비스가 해결하기 못한 서비스성 덕분 부하를 해결하였다. 또한 일반 프록시들의 정보를 유지하고 있는 디스패처가 해결하기도 하였던 사용자의 환경에 적응적으로 스트리밍 서비스를 가능하게 하여, 제안된 구조에 적용할 케이스 산업별 업계 및 교체 정책을 제시하여 케이스 공감의 효율성을 높였다.

우리는 보다 더 높은 서비스를 제공하기 위해 클러스터 간의 통신과 동기화 문제를 사용자에게 서비스 되는 디스크립션 수의 관통을 적게 하는 방법과 디스크립션의 조합을 서비스 하고 있는 동안 양계적으로 유지 하는 방법 등을 고안 중이다. 또한 제안된 협업 프록시 구조의 성능을 평가하기 위한 실험 단계에 있다.

참고문헌


