RTSP 기반의 동적 스트리밍 서버의 설계 및 구현

연재혁* · 임효택* · 박재홍**

*동서대학교 컴퓨터정보공학부, **(주)유라클

Design and Implementation of Dynamic Streaming Server based on RTSP

Jae-Hyuk Yeon* · Hyo-Taek Lim* · Jae-Hong Park**

*Division of Computer Engineering, Dongseo university, **Uracle Corporation

E-mail : cdopt@naver.com

요 약

본 논문에서 RTSP 기반의 스트리밍 서버를 구현해서 스트리밍 서비스가 가능하게 하였다. 클라이언트가 서버에 스트림을 요청하게 되면 그 때부터 주기적으로 클라이언트와 서버(웹 서버)간의 통신을 통해 전송 속도를 측정한다. 서버에서는 하나의 무비클립에 대해 분할 인코딩 해 놓은 스트림들을 전송 속도를 기반으로 차별 전송을 한다. 측정 속도에 따라 서버에서 보내는 스트림이 달라지며 스트림의 위치를 기억해서 달라진 전송 속도에 따라 다른 스트림을 보낼 때, 그 위치 그대로 보냄으로써 끊임없는 변환이 이루어진다.

ABSTRACT

This paper is to implement a RTSP based streaming server in order to provide streaming service. When a client requests a stream from a server, transmission speed is measured on a regular basis through communications between the client and the server. Server will send different encoded movie clip streams based on measured transmission speeds.

Different streams are sent to a server depending on the transmission speed and their locations are saved when streams are sent. Even though different streams are sent, the streams will be sent without any interruptions because they are sent from their original locations.

키워드

Dynamic, Streaming, RTSP, Network

Ⅰ. 서 론

스트리밍(Streaming)이란 1995년 리얼 네트워크 사가 개발한 리얼 오디오에서 처음 선보인 기술로 인터넷에서 음성이나 영상, 애니메이션 등을 실시간으로 재생하는 기법을 말한다. 현재 많은 포털 사이트와 기업들이 스트리밍 시스템을 개발 및 서비스 하고 있다. 대표적으로 야구, 축구 등의 생방송을 사용자에게 서비스 하고 있으며 기업들은 화상 회의나 실시간 컨퍼런스 등을 위해서 사용하고 있다. 스트리밍의 이점은 스트림이 모두 전송되기 전에도 클라이언트 브라우저나 플러그인이 데이터의 표현을 할 수 있어서 전송과 동시에 시청이 가능하며 하드 디스크 드라이브의 용량도 영향을 거의 받지 않는다. 그러나 스트리밍의 단점

을 살펴보면 비디오의 품질과 전송률에 대한 제어가 되지 않아서 인터넷 속도가 느린 사용자들은 끊김 현상이 발생한다. 이 점은 매우 치명적이며 스트리밍이 일부 인터넷 회선이 빠른 사람들만을 위한 서비스로만 인식되고 있는 경우가 있다.

현재 상용화 되어 있는 스트리밍 서버들은 Smooth, HTTP streaming등 여러 가지가 있지만 모두 전송률 제어가 불가능 하며 일부 서버는 비디오 품질 제어까지도 불가능하다. 이러한 점을 착안하여 본 논문에서는 스트리밍 서버에 요청하는 클라이언트마다 상태를 확인해서 차별화 된 스트리밍 서비스를 제공한다. 처음, RTSP 기반의 서버를 구현하고, 각각의 무비클립에 대해 여러 방법으로 인코딩 시킨 영상들을 준비한다. 샘플 영상은 서버에 100, 300, H-264 100, H-264 300kbit 정도로 준비하였다. 클라이언트에서 영상에 대한 요청이 들어오면 클라이언트의 대역폭(bandwidth)을 먼저 알아낸 다음 대역폭에 맞추어서 전달할 영상을 스트리밍 한다. 본 논문에서는 인코딩된 영상에 대해서 미리 준비를 해놓고 구현을 하였으며 차후에 인코딩 기법까지도 서버에서 실시간으로 처리를 할 수 있도록 할 예정이다.

		Smooth streaming	HTTP streaming	Dynamic streaming
지원 OS	Server	Windows vista 이상	Web server가 구동되는 OS	Windows, Linux, Mac OS
	Client	Windows, MAC OS (Silverlight 최신 버전 필요)	Mac OS , windows	Windows, Mac OS, Linux (flash가 동작하는 환경)
클라이언트 플랫폼		Web, desktop	Web, desktop	Web, desktop, mobile device
비디오 품질 제어		클라이언트 측에서만 가능 (Visual C#, Visual basic)	불가능	가능 (Action script 3.0)
전송률 제어		불가능	불가능	불가능

그림 1. 상용 스트리밍 서버들의 특징

II. 스트리밍 서비스의 구현

가. 서버 구현

서버는 C++ Language로 설계 하였으며 차후에 수정이 용이하도록 Class로 설계 하였다. Streaming 부분은 현재 Open Source로 배포중인 Live555를 이용하여 설계하였다. Live555는 RTSP, RTP등의 프로토콜을 지원할 수 있는 API와 그 외 데이터들의 전송 및 스트리밍에 관련된 API를 제공한다. 그리고 C++ Language로 설계가 가능하도록 되어있다. 서버의 구현 내용은 다음과 같다. 우리는 서버로 클라이언트가 연결을 요청하면 처음 서버 파일의 목록을 보내주고 주기적인 Server-Client간의 통신을 통해 Bandwidth를 측정한다. 측정된 Bandwidth를 토대로 클라이언트에서 원하는 무비 클립에 대해 Streaming 서비스를 수행한다.

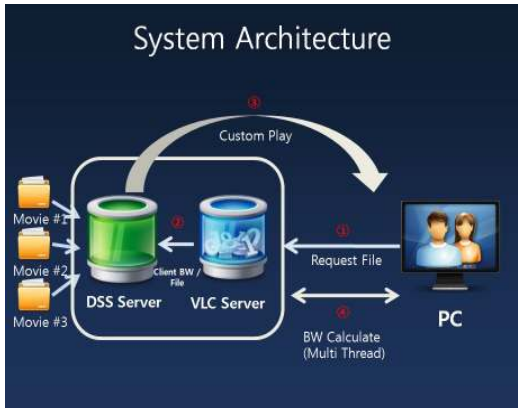

그림 2. Dynamic Streaming Architecture

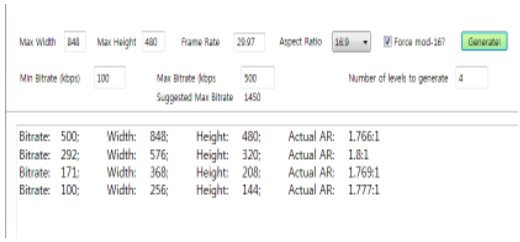

그림 3. Bandwidth Calculate

나 Bandwidth 계산

Server-Client의 Bandwidth 측정방법은 현재 Silverlight의 Smooth Streaming방식에서 사용하고 있는 계산법을 따른다. Bandwidth에 따른 인코딩 방식은 그림 3에서 설명하고 있으며 계산과정을 거친 뒤, 파일 서버로 요청 파일과 Client IP, Bandwidth를 전송한다.

다. 파일 서버 및 구현

파일 서버에서는 기존의 서버에서 전송된 정보 데이터들을 획득 한 뒤 저장되어 있는 인코딩 파일을 선택해서 보내게 된다. 대역폭에 따른 목록을 세부적으로 나눈 뒤 해당 대역폭에 있는 파일을 스트리밍 하게 된다. 파일을 스트리밍 하기 위해서는 VLC media player(이하 VLC)를 이용한다. VLC는 Open Source로 구성이 되어 있으며 제공하는 미디어 플레이어를 사용자가 SDK를 통해 Customize 시킬 수 있다. VLC SDK의 방대한 기능 중에서 Play, Pause, Setup, Volume등의 기본적인 스트리밍 관련 API만 따로 추출하여 사용하였다.

라. 클라이언트 구현

서버의 모든 과정을 거쳐서 클라이언트는 스트리밍 서비스를 받게 된다. 이 때 클라이언트는 VLC를 통해 스트리밍 서비스를 받게 되며 서버에서 전송된 스트림을 VLC에서 실제로 사용자에게 보여주기 위해서 각각의 무비클립의 포맷에 맞게 가공한다. 이를 위해서는 클라이언트 PC에 VLC Player가 설치되어 있어야 하며 우리는 VLC Player의 Core에 접근하여 Stream의 정보와 재생 형식, Bandwidth를 설정하여 Custom Streaming Player를 구동하게 된다. 클라이언트는 VLC를 통해 스트림의 가공이 이루어지므로 세부적으로 스트리밍의 구동방식을 전혀 이해 할 필요가 없으며 간단한 GUI 디자인을 통해 기본적인 Media Player의 느낌을 받게 된다.

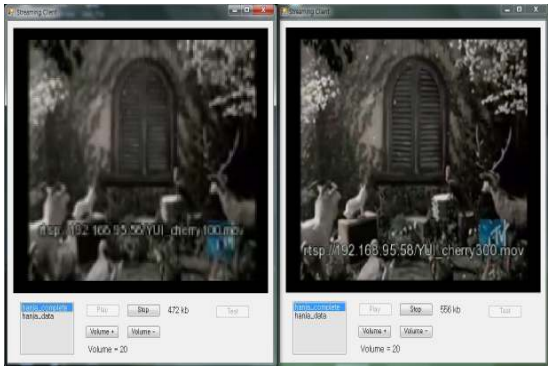

그림 4. Bandwidth에 따른 스트리밍 제공 영상의 차이 좌- 100kb, 우- 300kb

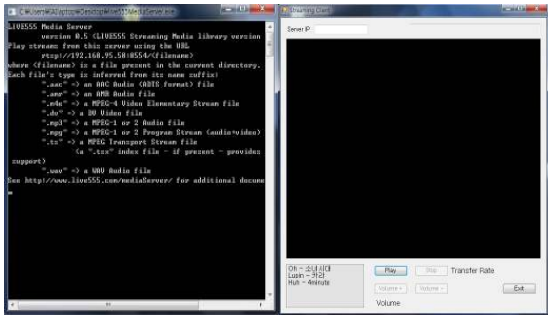

그림 5. 좌 - Live555 Server,
우-GUI, Server Connect를 포함한 버전

Ⅲ. 측정 결과 및 차이점

측정 결과를 확인하기 위해 무비클립은 4분 내외의 뮤직비디오 4개를 사용하였고 각 무비클립마다 100, 300, 500kb의 인코딩으로 나누어 서버에 준비하였다. 서버는 Pentium 4, Windows XP의 환경을 구축하고 WIFI, Lan의 두 가지 환경에서 테스트를 하였다.

측정 결과, 두 가지 테스트의 공통점은 대역폭의 레벨이 변경이 되면서 서버에서 인코딩을 바꾸기 때문에 잠시 동안 블랙 스크린이 나타나게 된다. 블랙 스크린은 클라이언트에서 보여주는 panel의 기본 컬러 값이 검은색이기 때문에 스트림이 바뀌는 순간에 보여 진다. 아주 짧은 순간이지만 변화에 민감한 사용자들을 위해서는 인코딩이 바뀌기 전에 무비클립의 마지막 프레임을 기록해 두었다가 바뀐 무비클립을 재생하기 전까지 스크린에 display 시켜야 한다. 그렇게 한다면 이질감이 많이 사라 질 것이다.

그 외 각각의 차이점은 다음과 같다. 먼저 Lan의 경우는 대역폭이 크게 변화되는 순간이 없기 때문에 강제로 폭을 낮추어 바꾸어도 부드럽게 진행이 된다. 하지만 WIFI를 장착한 모바일 디바이스, 노트북 같은 경우는 기본적인 전송 속도가 Lan에 비해서 많이 낮기 때문에 대역폭을 강제로 조절해서 폭을 자주 바꾸게 되면 화질이 바뀌는 폭도 역시 많아진다. 모바일 디스플레이는 특성상 화면이 작기 때문에 별 다른 이질감은 느껴지지 않지만, 노트북의 경우는 자주 화질이 바뀌는 모습을 느낄 수 있어서 조금은 이질감이 생긴다. 이 현상을 해결하기 위해 처음 구현했던 대역폭이 바뀌는 구간마다 무비 클립의 인코딩을 바꾸는 방법 보다 일정 시간별로 평균 대역폭의 구간으로 무비 클립의 인코딩을 변경해주면 더욱 효과적일 것이다.

Ⅳ. 결 론

본 논문에서는 현재 일반적으로 서비스 되고 있는 스트리밍 서비스의 단점을 극복 할 수 있는 구현 방법을 제시하였다. 기존 기술과의 차이점을 간단하게 설명하자면 다음과 같다.

첫 번째, 순수 Open Source를 이용한 구현으로 쉽게 customize가 가능하다. 현재 상용화 되고 있는 여러 스트리밍 서버들은 제조사가 개발한 대로 사용해야 하며 기본 설정 이외에는 고치거나 수정할 부분이 있더라도 수정할 수 없다. 하지만 Open Source로 개발한 본 논문의 스트리밍 서비스는 어느 누구나 수정이 가능하며 필요한 기능들을 개발 문서를 통해 참고하여 확장 및 업데이트가 가능하다.

두 번째, 가격적인 면에서 이익이며 전송률 제어가 가능하다. 하드웨어 인코더를 이용하면 스트리밍 중 영상품질 및 전송률을 제어할 수 있으나 가격이 비싸다. 그리고 스트리밍 서버를 이용하면 전송률을 제어할 수 없으며 HTTP streaming 서버 같은 경우는 클라이언트의 종류에 따른 인코딩마저 불가능하다. 본 논문의 스트리밍 서비스는 전송률 제어가 쉽게 가능하며 사용자의 입맛에 맞게 전송률에 따른 인코딩 파일도 선택할 수 있으며 가격의 문제도 없다.

본 논문의 스트리밍 서비스는 새로운 기능 외에 기존의 기능은 시중의 상용 스트리밍 방식보다 구성이 덜 되어 있다. 오픈 소스로 구성을 한 만큼 그런 기능들을 많은 개발자들이 협력을 통해 추가 구현을 해야 하며 Live Streaming Server를 추가적으로 구현 할 경우 실시간 인코딩 기법의 연구가 필요하다.

참고문헌

[1] Joe Follansbee, Hands-On Guide to Streaming Media (An Introduction to Delivering On-Demand Media), FOCAL PRESS, 2006

[2] Microsoft Corporation, Silverlight streaming, http://silverlight.live.com/

[3] VideoLan Group, VLC media player SDK, http://www.videolan.org/developers/

[4] 박동재, 성재득, 이동수외 3명, 분산 클러스터링을 이용한 네트워크 동영상 스트리밍, 한국 정보 과학회, 2007