리눅스 서버를 이용한 동영상 데이터 실시간 스트리밍 서비스 연구

장승주* · 허원영** · 유현민** · 이창훈** · 신우호**

A Study of Real-Time Video Streaming Data Service on the Linux Server

Seung-Ju Jang* · Heo Won-Yeong** · Yoo Hyun-Min** · Lee Chang-Hoon** · Shin Woo-Ho**

이 논문은 KAIST 부설 한국과학영재학교 R&E 연구의 성과물로 2012년도 한국창의재단
(과학기술진흥기금/복권기금)의 지원을 받아 수행되었습니다.

요 약

본 논문은 안드로이드 시스템 환경에서 리눅스 서버 시스템을 통하여 실시간으로 동영상을 스트리밍 서비스하는 시스템을 제안한다. 본 연구에서 개발한 어플리케이션으로 동영상을 촬영함과 동시에 스트림 데이터를 생성하고, 이 데이터를 Apache Tomcat 서버의 MultipartRequest 클래스를 이용하여 리눅스 서버로 전송하는 기능을 구현하였다. 또한, 안드로이드의 비디오 플레이어와 미디어 플레이어 클래스를 이용하여 실시간으로 동영상 스트림 데이터를 비디오 플레이어에서 재생하는 방식과 캐시(cache)에 스트림 데이터를 저장함과 동시에 미디어 플레이어를 이용하여 재생을 하는 방법을 개발하였다. 본 논문에서 제안한 시스템 구조와 어플리케이션이 설계된 실험 과정을 통한 실험의 결과, 모두 정상적으로 동작이 된다는 것을 확인 할 수 있었다.

ABSTRACT

This paper suggests a method of developing live media streaming service through Linux server system on android system environment. The android application constructed in the experiment is able to record media while sending it to Linux server. Generated real time media data is send to linux server through Multipart Request class of Apache Tomcat server constructed on Linux system. Also in this research, by utilizing Android video player and media player class, development of android application structures was accomplished, which has methods of; playing live media stream data on video server, or playing live media stream data while saving stream data in cache. The structure and function of suggested system and application is confirmed by series of experiments.

키워드

스트리밍, 리눅스 서버, 안드로이드, 실시간, 아파치 톰캣, 캐시 파일

Key word

Streaming, Linux Server, Android, Real-Time, Apache Tomcat, Intent, Cache File

---

* 정회원 : 동의대학교 컴퓨터공학과 교수(sjjang@deu.ac.kr)
** 준회원 : 한국과학영재학교 학생

접수일자 : 2012. 11. 12
심사완료일자 : 2013. 04. 07

Open Access <http://dx.doi.org/10.6109/jkiice.2013.17.4.893>

©This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(<http://creativecommons.org/licenses/by-nc/3.0/>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Copyright © The Korea Institute of Information and Communication Engineering.

I. 서 론

일반적으로 웹상의 미디어 데이터를 전송하고 재생하는 데에 사용되는 기술에는 다운로드(Download) 방식, 점진적 다운로드(Progressive Download) 방식, 그리고 스트리밍(Streaming) 방식으로 모두 세 가지가 있다. 이 중에서도 스트리밍 기술은 미디어 데이터를 전송받는 동시에 이를 시청할 수 있다는 점에서 각광받고 있다. 스트리밍 기술은 1995년 마이크로소프트(Microsoft)사의 ActiveMovie 미디어 플레이어와 Real Networks사의 야구 경기 스트리밍 생중계 방식에 의하여 본격적으로 상용화되기 시작하였으며 현재까지 사용되고 있다 [1, 2].

안드로이드 운영체제의 스마트폰이나 스마트패드 등을 포함한 이동 통신 기기가 국내외의 다양한 분야에서 널리 사용되고 있어 적용률이 높다[3]. 이 점에 비추어 볼 때 안드로이드 시스템 환경에서 스트리밍 기술은 유망한 분야지만, 아직 깊게 연구되지는 않았다.

본 논문에서는 안드로이드 환경에서 실시간으로 동영상 데이터를 스트리밍 서비스하는 것에 대해 설명할 것이다. 본 논문의 구체적인 연구 내용은 다음과 같다. 첫째, 안드로이드 시스템에서 동영상을 촬영하여 리눅스 서버로 데이터를 실시간으로 보내는 어플리케이션을 설계하였다. 둘째, 리눅스 환경에서 스트리밍 서버를 구축하여 첫째 단계의 안드로이드 환경에서 보내는 스트림 데이터를 실시간으로 수신하고 추가적으로 스트리밍을 가능하게 하는 환경을 구축하였다. 셋째로, 안드로이드 환경에서 스트리밍 서버의 스트림 데이터를 재생할 수 있는 어플리케이션 개발 및 첫째 단계에서 언급한 어플리케이션과의 통합을 진행하였다.

본 논문은 ‘야외에서의 IP 카메라 어플리케이션을 통한 CCTV’와 ‘안드로이드 클라이언트와의 연결 및 감시, 안드로이드를 통한 실시간 미디어 방송’, ‘안드로이드 상에서의 실시간 스트리밍’ 등을 참고하였다 [4, 5].

본 연구에서 리눅스 서버 환경과 안드로이드 클라이언트 환경에서 실시간으로 동영상 촬영 및 전송이 이루어지는지 시험해 보았으며 이러한 내용을 5장에 걸쳐 설명할 것이다. 본 논문의 2장은 시스템 구조, 3장은 본 연구에서 개발한 어플리케이션의 전송 방식, 4장은 어

플리케이션의 수신 방식, 5장은 어플리케이션을 이용한 실험, 6장은 결론 및 향후 연구에 대해서 설명할 것이다.

II. 본 논문에서 제안하는 시스템 구조

본 장에서는 본 연구에서 제안하고 구축한 시스템에 대하여 설명 할 것이다. 아래의 그림 1은 본 논문에서 제안하는 시스템의 전체구조이다.

그림 1. 본 논문에서 제안하는 시스템의 전체구조
Fig. 1 Suggested System Structure

본 연구에서 최종 목표로 하여 구축한 시스템의 전체 구조는 그림 1과 같다. Android User Group 1은 안드로이드 카메라를 통해 동영상 스트림 데이터를 실시간으로 생성하고 리눅스 서버로 해당 스트림 데이터를 전송하는 클라이언트이다. 중앙의 Linux Server System은 Android User Group 1에서 보낸 스트림 데이터를 받아 Android User Group 2로 보내는 서버이다. Android User Group 2는 Linux Server System에서 보내진 스트림 데이터를 실시간으로 받아 재생하는 클라이언트이다.

안드로이드 카메라에서 리눅스 서버 시스템으로 카메라 출력 스트림 데이터를 전송하는 어플리케이션과 리눅스 서버 시스템으로부터 스트림 데이터를 수신하고 재생할 어플리케이션이 필요한데, 본 연구에서는 두 개의 어플리케이션을 통합하여 개발하였다. 다음 그림 2는 본 연구에서 개발한 어플리케이션에서 리눅

스 서버 시스템으로 스트림 데이터를 전송하는 시스템 구조이다.

Android Clients (Video Stream Sender)

Linux Server System
with Apache Tomcat "MultipartRequest"

그림 2. 어플리케이션
Fig. 2 Sending Method for Application

본 연구에서 개발한 어플리케이션으로 동영상을 촬영하면 카메라 출력 스트림 데이터를 생성하여 리눅스 서버 시스템으로 보내게 된다. 리눅스 서버 시스템에서는 수신한 스트림 데이터들을 Apache Tomcat 서버의 MultipartRequest 클래스를 이용하여 저장하게 된다.

다음 그림 3은 본 연구에서 개발한 어플리케이션에서 리눅스 서버 시스템으로부터 스트림 데이터를 전송 받아 재생하는 시스템 구조이다.

그림 3. 어플리케이션의 수신 방식
Fig. 3 Receiving Method for Application

어플리케이션을 이용하여 리눅스 서버 시스템에 저장되어 있던 동영상 데이터 스트림 데이터를 수신 받아 재생할 수 있다.

III. 어플리케이션의 스트림 전송 방식

본 장에서는 본 연구에서 개발한 어플리케이션에서 리눅스 서버 시스템으로 스트림 데이터를 실시간으로 전송하는 시스템의 알고리즘을 세부적으로 설명한다. 다음 그림 4는 알고리즘의 작동 구조를 도식화한 것이다.

그림 4. 동영상 스트림 생성 및 전송 알고리즘
Fig. 4 Media Stream Generation and Sending Algorithm

리눅스 서버 시스템으로 스트림 데이터를 전송하는 어플리케이션은 세 개의 메소드와 한 개의 클래스를 사용하며, 각 메소드 및 클래스에 대한 세부적인 설명은 다음과 같다.

3.1. OnCreate(메소드)

본 어플리케이션을 불러 왔을 때 실행되는 메소드로, 레이아웃에서 타이틀바를 없애고 전체화면으로 보이도록 변경하며, 촬영 중 화면이 꺼지지 않도록 설정한다. 또한 Recording이라는 이름의 Button과 동영상 미리보기 표면으로 사용할 SurfaceHolder를 선언한다.

3.2. Recording.setOnClickListener(메소드)

위의 3.1절에서 언급한 Button인 Recording이 클릭되었을 때 실행되는 메소드이다. 촬영 중이 아닌 경우 새로운 MediaRecorder 객체를 선언하여 동영상 및 오디오 공급원, 출력 포맷, 인코딩 코덱과 파일 저장 경로 등을 지정한다. 그리고 동영상 미리보기 표면을 SurfaceHolder로 설정하는 과정을 거친 후 녹화를 시작한다. 반대로 촬영 중인 경우 방금까지의 촬영을 정지한 후 사용되고 있던 MediaRecorder 객체를 해제한 뒤에 HttpFileUpload 메소드를 불러 파일 전송을 시작한다.

3.3. HttpFileUpload(메소드)

HttpConnection 클래스를 새로 선언하여 시작한다.

3.4. HttpConnection(클래스)

안드로이드 운영체제 3.0 버전인 허니콤(Honeycomb)부터는 HTTP를 이용한 네트워킹이 메인 스레드에서 처리될 수 없도록 설정되었으므로 스레드 하나를 생성하여 시작한다. Apache Tomcat 서버에 MultipartRequest를 이용하여 Stream을 받아 파일로 저장하도록 되어 있는 jsp 파일이 있어야 한다. FileInputStream를 선언하고, HttpURLConnection를 선언한 뒤에 openConnection 메소드를 이용하여 해당 jsp 페이지에 접속한다. HTTP 통신 연결 관련설정을 거친 뒤에 DataOutputStream를 선언했던 HttpURLConnection 타입의 변수에 getOutputStream 메소드를 사용하여 선언한다. 이후 buffer를 이용하여 전송할 동영상의 바이트가 더 이상 남아 있지 않을 때까지 write(buffer)로 계속 스트림을 보내준다. 모든 바이트를 전송하였으면 FileInputStream을 닫는다. 마지막으로 DataOutputStream 변수에 flush 메소드를 이용하여 파일 전송을 완료하고, 이 스트림도 close 메소드로 닫는다.

IV. 어플리케이션의 스트림 수신 방식

본 장에서는 본 논문에서 개발한 어플리케이션이 작동하는 세부적 흐름에 대하여 설명할 것이다. 본 논문에서는 두 가지 방식으로 동영상 스트림 데이터를

실시간으로 수신하는 어플리케이션을 개발하였다. 하나는 안드로이드에 내장되어 있는 비디오 플레이어를 Intent하여 데이터를 받는 방식이고, 다른 하나는 안드로이드의 캐시 파일을 이용하여 실시간으로 시청하는 방식이다.

4.1. 비디오 플레이어 Intent 어플리케이션

본 절에서는 비디오 플레이어를 이용하여 동영상 스트림 데이터를 수신하는 어플리케이션의 세부 흐름도에 대해서 설명할 것이다. 아래의 그림 5는 비디오 플레이어를 Intent하여 동영상을 재생하는 어플리케이션의 작동 순서를 나타낸 것이다.

그림 5. 비디오 플레이어 intent 어플리케이션 순서도
Fig. 5 Flow Chart of Video Player Intent Application

어플리케이션을 시작하면 OnCreate 메소드가 실행되고 Activity가 시작된다. 이후 EditText, WebView, 그리고 Button 2개를 선언한다.

WebView는 서버에 저장되어 있는 HTML 파일의 URL을 지정하여 그 파일의 내용이 보이게 한다. 이 HTML 파일에는 서버에 저장되어 있는 영상의 목록을 가지고 있다. 그 영상의 제목을 EditText에 입력하여 해당 영상을 지정할 수 있다.

‘시청’ 버튼을 클릭하면 EditText에 입력되어 있는 영상의 URL로 연결하여 Intent한다. 이 때 EditText는 안드로이드에 내장되어 있는 MediaPlayer 내에 들어 있다.

4.2. 캐시 파일을 이용한 스트림 수신 어플리케이션

본 절에서는 캐시 파일을 이용하여 동영상 스트림 데

이터를 실시간으로 수신하는 어플리케이션에 대해서 설명할 것이다.

이 어플리케이션을 실행하면 **OnCreate** 함수가 시작되면서 **playButton**이라는 이름의 **Button** 하나가 선언된다. 이 버튼을 클릭하면 **StreamingMediaPlayer** 클래스가 선언되면서 서버의 동영상과 **URL**을 연결하여 영상 재생을 시작한다.

StreamingMediaPlayer 클래스가 시작되면 **download AudioIncrement**가 실행되어 캐시 영역에 동영상 스트림 데이터를 다운로드 받기 시작한다. 동시에 **testMediaBuffer** 메소드가 실행이 되고, 이때 일정 용량 이상 다운로드 받으면 **startMediaPlayer** 메소드가 실행된다. 이 메소드 안에서는 스트림 데이터를 재생시키는 **createMediaPlaer** 메소드를 사용하여 영상을 재생한다. 이 때 **createMediaPlayer** 메소드는 안드로이드에 내장되어 있는 **MediaPlayer** 클래스를 참조한다.

동영상을 다운로드 받으면서 재생을 하기 때문에 재생 시작 후에 새로 받은 파일로 인하여 재생이 끊어질 수 있다. 이를 방지하기 위하여 **transferBufferToMediaPlayer** 메소드는 기존 영상 파일과 새로 받은 영상 파일을 통합하여 재생이 끊어지는 것을 방지한다.

V. 실 험

본 장에서는 본 연구에서 개발한 어플리케이션으로 안드로이드 시스템에서 촬영한 동영상 데이터를 스트림 데이터 형태로 리눅스 서버 시스템을 통하여 다른 안드로이드 시스템으로 전송하고, 재생한 결과를 사진으로 보여 주고자 한다. 본 연구에서는 아래와 같은 환경에서 개발 및 실험을 하였으며, 정상적으로 작동하는 것을 확인하였다.

1) 리눅스 서버 시스템 환경

Ubuntu 12.04 + Apache 2.4 + Tomcat 7.0.29

2) 안드로이드 클라이언트 환경

삼성전자 애니콜 SHV-E120S (갤럭시S2 HD LTE, Android 4.0), LG전자 LG-LU3700(옵티머스 원, Android 2.2)

5.1. 동영상 촬영 및 전송

다음 그림 6은 어플리케이션을 실행하고 촬영을 하기 전의 화면이다.

그림 6. 동영상 촬영 전 화면
Fig. 6 Screen Before Media Record

어플리케이션이 실행이 되면 **OnCreate** 메소드를 거쳐 레이아웃이 설정된 상태가 된다. 그림 6의 우측 상단의 촬영 버튼을 클릭하면 촬영이 시작된다. 다음 그림 7은 동영상을 촬영하고 있는 화면이다.

그림 7. 동영상 촬영 화면
Fig. 7 Screen of Recording Media

어플리케이션으로 촬영한 동영상은 촬영하고 있는 안드로이드 클라이언트에서 **SD** 카드 내부의 한 폴더에 임시로 저장하게 된다. 그림 7의 우측 상단의 중지 버튼을 클릭하게 되면 촬영이 정지 된다. 다음 그림 8은 촬영이 중지되고 동영상 데이터를 리눅스 서버 시스템으로 전송을 완료한 화면이다.

그림 8. 동영상 데이터 전송 완료 화면
Fig. 8 Screen of Succeeded Media Data Sending

중지 버튼을 클릭하게 되면 동영상 촬영이 종료되고, 3.3과 3.4절에서 설명한 일련의 과정을 거쳐 리눅스 서버 시스템으로 동영상 데이터를 업로드 한다. 이 경우 반드시 인터넷이 연결되어 있어야만 한다.

5.2. 비디오 플레이어 Intent 어플리케이션

다음 그림 9는 비디오 플레이어를 이용한 어플리케이션을 실행한 뒤의 화면이다.

그림 9. 비디오 플레이어 intent 어플리케이션 시작 화면
Fig. 9 Screen of Video Player Intent Application Start

EditText와 두 개의 버튼 밑에 있는 빈 화면은 WebView로서, 서버에 저장된 동영상 파일의 리스트가 나오는 부분이다. 이 파일 리스트를 보면서 그림 9와 같이 보고 싶은 동영상의 비밀번호를 입력할 수 있도록 하

였다. ‘비밀번호 입력’이라고 쓰여 있는 EditText 부분에 비밀번호를 입력한 뒤에 ‘시청’ 버튼을 클릭하게 되면, 해당 비밀번호를 파일명으로 하는 동영상을 서버에서 찾아 재생하게 된다. 이 경우, 제대로 된 비밀번호를 입력하지 않고서 ‘시청’ 버튼을 클릭하게 되면 ‘재생할 수 없는 동영상입니다.’라는 오류가 발생하게 된다.

비밀번호 입력창에 촬영 중에 주어진 비밀번호를 입력하여 재생 버튼을 클릭하면 동영상이 재생되게 된다. 이러한 동영상 재생 방법은 안드로이드 운영체제에 내장된 비디오 플레이어 어플리케이션을 불러와서 사용하게 된다.

5.3. 캐시 파일을 이용한 스트림 수신 어플리케이션

다음 그림 10은 캐시 파일을 이용하여 동영상 스트림 데이터를 수신하는 어플리케이션의 화면이다. 오른쪽에 보이는 재생 버튼을 누르면 서버에 있던 영상의 URL로 연결하여 영상을 재생한다.

그림 10. 캐시 파일을 이용한 어플리케이션 화면
Fig. 10 Screen of Application Using Cache File

이 어플리케이션과 5.2의 어플리케이션의 차이점을 비교하면, 전자는 어플리케이션 내에서 화면을 제공하여 영상을 재생하는 반면에, 후자는 비디오 플레이어를 Intent한 것으로 어플리케이션 내에서 재생하는 것이 아니다.

위와 같은 방식으로 작동하는 동영상 데이터 실시간 스트리밍 전송 어플리케이션의 성능을 측정하기 위하여, 본 논문에서는 총 10회에 걸쳐 촬영이 완료된 동영상 데이터를 서버로 전송하여 서버에서 전체 데이터를 받아들이는 데에까지 소요되는 시간을 측정하였다. 인터넷 환경으로는 WI-FI를 이용하였다. 그 결과는 아래의 표 1과 같았다.

표 1. 전송 어플리케이션 성능 측정 결과
Table. 1 A Result of Performance Test for Sending Application

시도	소요시간(ms)	시도	소요시간(ms)
1차	2115	6차	2063
2차	2060	7차	2213
3차	2031	8차	2074
4차	2051	9차	2061
5차	2040	10차	2058
평균 소요시간(ms)		2076.6	

총 10회 중 가장 적은 시간이 소요된 경우에는 2031ms(2.031초), 가장 많은 시간이 소요된 경우에는 2213ms(2.213초)가 소요되었으며, 평균 소요시간은 2076.6ms(2.076초가량)으로, 거의 2초에 가까운 시간 내에 전송이 완료됨을 확인했다. 각 소요시간마다 큰 편차는 없이 2초에서 2.2초가량의 시간 내로 촬영한 동영상이 전송되는 것을 확인할 수 있었다.

본 논문에서 제작한 어플리케이션에서는 동영상을 5초 단위로 잘라 촬영하고 전송하며, 전송을 시작함과 동시에 다음 동영상을 촬영하기 시작하므로 정상적인 인터넷 환경에서는 끊기지 않는 실시간 동영상 스트리밍 데이터 전송이 가능하다. 또한, 인터넷 환경이 좋지 않더라도 정상적인 인터넷 환경의 50% 정도가 되는 속도를 유지할 수 있다면 충분히 큰 문제가 없이 실시간 동영상 스트리밍 데이터 전송을 할 수 있다. 더불어 서버에서의 데이터 수신 성공률은 10회 모두 100%로, 데이터를 완전하게 전송하고 수신하는 데에 어려움이 없음을 확인할 수 있었다.

다음으로 동영상 데이터 실시간 스트리밍 재생 어플리케이션의 성능을 측정하기 위하여, 본 논문에서는 총 10회에 걸쳐 촬영이 완료되어 서버로 전송된 스트리밍 데이터를 안드로이드 클라이언트 측에서 전송받는 데에까지 소요되는 시간을 측정하였다. 인터넷 환경으로는 WI-FI를 이용하였다. 그 결과는 아래의 표 2과 같았다.

총 10회 중 가장 적은 시간이 소요된 경우에는 2041ms(2.041초), 가장 많은 시간이 소요된 경우에는 2228ms(2.228초)가 소요되었으며, 평균 소요시간은 2114.9ms(2.076초가량)으로, 거의 2초에 가까운 시간 내에 수신이 완료됨을 확인했다.

표 2. 재생 어플리케이션 성능 측정 결과
Table. 2 A Result of Performance Test for Playing Application

시도	소요시간(ms)	시도	소요시간(ms)
1차	2097	6차	2100
2차	2103	7차	2135
3차	2228	8차	2064
4차	2041	9차	2191
5차	2076	10차	2114
평균 소요시간(ms)		2114.9	

각 소요시간마다 큰 편차는 없이 2초에서 2.2초가량의 시간 내로 서버에 있는 동영상 스트림이 수신되는 것을 확인할 수 있었다. 본 논문에서 제작한 어플리케이션에서는 동영상을 5초 단위로 잘라 촬영하고 전송하며, 이렇게 서버에 전송된 동영상 스트림을 수신 받아 재생하므로 정상적인 인터넷 환경에서는 끊기지 않는 실시간 동영상 스트리밍 데이터 수신이 가능하다. 또한, 인터넷 환경이 좋지 않더라도 정상적인 인터넷 환경의 50% 정도가 되는 속도를 유지할 수 있다면 충분히 큰 문제가 없이 실시간 동영상 스트리밍 데이터 수신을 할 수 있다. 더불어 클라이언트에서의 데이터 수신 성공률은 10회 모두 100%로, 데이터를 완전하게 전송하고 수신하는 데에 어려움이 없음을 확인할 수 있었다.

VI. 결 론

본 논문에서는 최근 안드로이드 운영체제를 탑재한 스마트폰의 점유율이 비교적 높은 편이므로 사용자들이 가장 쉽게 접근 가능한 스마트폰 운영체제이라는 점에서 안드로이드 시스템 기반 어플리케이션을 개발하였다. 또한, 오픈 소스 기반으로 접근성이 높으며 비용과 사양의 요구 사항이 높지 않고 보안성이 뛰어난 리눅스 환경의 서버를 사용하였다. 구체적으로 Apache Tomcat 서버를 설치한 뒤에 해당 서버의 MultipartRequest 클래스를 이용하여 안드로이드 클라이언트에서 동영상을 촬영하였고, 이를 리눅스 서버로 업로드 하는 방법을 이용하였다. 이를 통해 안드로이드 클라이언트에서 리눅스 서버로 동영상을 스트림 데이터의 형태로 전송할 수 있었으며, 안드로이드 클라이언트에서 리눅스 서버로

전송되어 오는 스트림 데이터를 재수신하여 재생할 수 있는 것을 확인하였다.

특히, 본 논문에서 제안하는 동영상 데이터 실시간 스트리밍 서비스에서는 Apache Tomcat 서버에서 사용할 수 있는 MultipartRequest 클래스를 이용하였다. 이 클래스는 파일 스트림을 통하여 서버에 파일을 저장하는 데에 주로 이용된다. 기존에 제안되어온 동영상 데이터 실시간 스트리밍 서비스 시스템에서는 동영상 자체의 코덱 등에 관한 처리를 필요로 하였으나, 본 논문에서 제안하는 시스템은 파일 전송 파일 스트림 자체를 전송하여 저장하는 MultipartRequest 클래스를 이용함으로써 해당 과정이 불필요하다. 또한, 안드로이드 API에서 기본적으로 제공하는 클래스와 Apache Tomcat 서버의 MultipartRequest만을 사용하여 다른 방면에서 활용할 때에도 간편하게 적용시킬 수 있다는 장점이 있다.

차후 보완 연구를 통하여 미디어 데이터의 실시간 스트리밍 기술을 속도와 안정성 측면에서 보완할 예정이다.

참고문헌

- [1] 김경희, “RTSP 기반 실시간 스트리밍 서비스 구현”, 안양대학교 첨단산업기술대학원 컴퓨터 정보통신학과, 2001
- [2] Apple, HTTP Live Streaming Overview, 2011
- [3] 삼성경제연구소, “스마트폰이 열어가는 미래”, CEO Information 741호, 2010
- [4] 김현영, “방송 및 SNS를 연계한 모바일 스트리밍 시스템”, 금오공과대학교 대학원 컴퓨터 공학과, 2011
- [5] 김종근, 최황규, “리눅스 클러스터 서버 상에서 RTSP 기반의 실시간 및 VOD 스트리밍 시스템 개발”, 정보 통신 논문지, vol. 7, 2003
- [6] 정선철, 반태학, “HTTP상에서 동적 적응적 스트리밍 시스템 구현”, 한국정보통신학회, Vol.16 No.3, 2012
- [7] 김대영, “최신 인터넷 스트리밍 기술동향과 시스코 콘텐츠 전달 솔루션”, 시스코 시스템즈 코리아, 2012
- [8] 박정현, 김동철, 이정민, 정광수, “스트리밍 서비스의 품질 향상을 위한 적응적 미디어 데이터 전송 제어 기법”, 한국정보과학회, Vol.39 No.2, 2012.
- [9] 임재걸, 이경훈, “동영상 재생 안드로이드 프로그램 구현”, 한국컴퓨터정보학회 학술발표논문집 Vol.2011, 2011
- [10] 권오병, 김경수, “FlashVideo를 적용한 인터넷 방송 동영상 솔루션의 설계 및 구현”, 디지털정책연구, Vol.16 No.6, 2012
- [11] 강미연, 주대연, 이선주, 김도완, 김재훈, 정원호, “IP 카메라 기반의 실시간 웹캐스팅 시스템”, 한국정보과학회 학술발표논문집, Vol.34 No.1D, 2007.

저자소개

장승주(Jang, Seung Ju)

1985년 부산대학교
계산통계학(전산학) 학사
1991년 부산대학교
계산통계학(전산학) 석사

1996년 부산대학교 컴퓨터공학 박사
1987년 ~ 1996년 한국전자통신연구원(ETRI) 시스템 SW연구실
1993년 ~ 1996년 부산대학교 시간강사
2000년 ~ 2002년 Univ. of Missouri at Kansas City, visiting professor
1996년 ~ 현재 동의대학교 컴퓨터공학과 교수
※관심분야: 운영체제, 임베디드 운영체제, 분산 시스템, 시스템 보안, 스마트폰 시스템 운영체제

허원영(Heo Won-Yeong)

2011년~ 한국과학영재학교 재학
※관심분야: 인공지능, 스마트폰 시스템 운영체제, 스마트폰 어플리케이션, 정보보안기술

유현민(Yoo Hyun-Min)

2011년~ 한국과학영재학교 재학
※관심분야: 게임 및 가상현실,
스마트폰 어플리케이션

이창훈(Lee Chang-Hoon)

2011년~ 한국과학영재학교 재학
※관심분야: 인공지능, 스마트폰
어플리케이션

신우호(Shin Woo-Ho)

2011년~ 한국과학영재학교 재학
※관심분야: 스마트폰
어플리케이션