ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

프레임 버퍼를 이용한 매끄러운 멀티미디어 스트리밍 서비스를 지원하는 안드로이드 앱 개발☆

Development of Android App for Supporting Smooth Multimedia Streaming Service Using Frame Buffer

서 상 민[1] 권 준 호[2] 최 윤 호[3*]
Sang-min Seo Jonnho Kwon Yoon-Ho Choi

요 약

본 논문에서는 기존의 안드로이드 화면 녹화 어플리케이션(이하, 앱)이 가지고 있는 녹화 기능의 코덱, 안드로이드 운영 체제 버전에 종속적인 특성과 별도의 데스크톱 PC와 유선으로 연결하여 서비스를 제공해야 하는 단점을 극복하는 새로운 앱을 제안한다. 또한 녹화된 화면 정보를 미디어 파일 포맷인 플래시 비디오(flash video) 파일을 이용하여 스트리밍 서비스를 제공하는 방법을 제안한다. 실험을 통해 제안한 앱 구현 방법의 경우 기존의 안드로이드 API를 이용하는 방법과 비교해 부호화 방법의 변경 없이 초당 20프레임 이상의 화면 녹화를 통해 매끄러운 영상을 녹화할 수 있음을 실험적으로 확인하였다.

☞ 주제어 : 모바일 장치, 안드로이드 앱, jni, 프레임 버퍼, 스트림 서비스

ABSTRACT

Existing Android applications for streaming video in real time are dependent on the codec, which composes the encoding function, and the version of Android operating system. Also, for streaming video in real time, most applications should be connected with a separate desktop PC. To overcome these disadvantages, we propose a new application, which records and streams video in real time. Specifically, the proposed application uses the flash video file format, which is the common media file format supported by various versions of Android operating system. Through experiments, we show that it is possible for the proposed application to record the video screens more than 20 frames per second and to stream it in real time while using the existing video encoding methods.

☞ keyword : mobile device, android app, jni, frame buffer, stream service

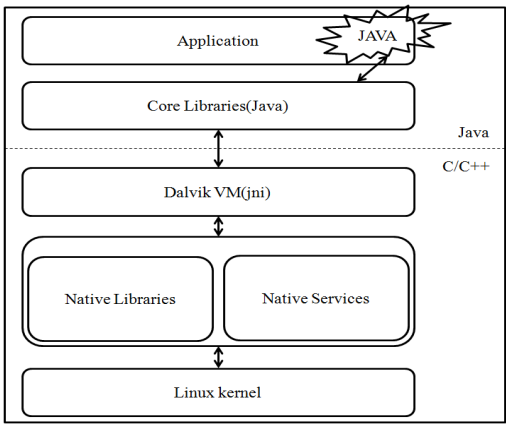
1. 서 론

모바일 네트워크와 더불어 모바일 디바이스에서의 하드웨어 성능 향상은 일반 텍스트가 아닌 음성 및 영상 데이터 같은 여러 가지 형태의 멀티미디어 정보를 다루는 앱의 개발을 촉진하고 있다[1]-[5]. 최근 들어 다양한 모바일 운영 체제가 개발되고 스마트 폰을 비롯한 다양한 모바일 디바이스가 일반 PC와 유사한 연산 성능과 기억 장치 용량 및 I/O 속도를 갖추게 되면서, LTE-A와 같은 네트워크의 대역폭을 증가시키는 링크 계층의 기술 발전과 더불어 네트워크상에서 높은 대역폭을 이용하는 비디오 화상 회의(video conferencing)와 VOD(Video-On-Demand) 등의 음성 녹음, 비디오 화면 녹화 및 이를 라이브 스트림(live stream)으로 서비스하는 모바일 화면 녹화 애플리케이션(이하, 앱(app))이 개발되고 있다.

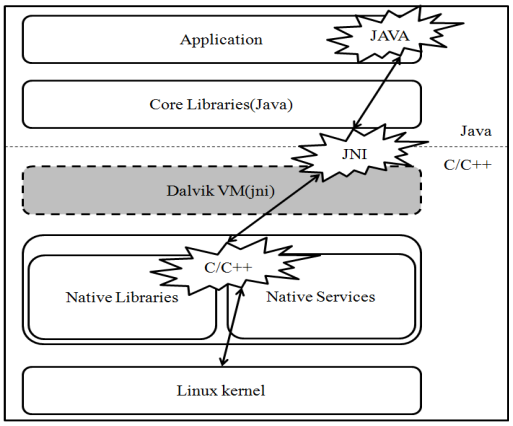
모바일 장치에서 음성 및 영상을 처리하는 앱은 기능에 따라 크게 영상 녹화 모듈과 이를 네트워크를 통해서 스트리밍으로 전송하는 스트리밍 모듈로 구성된다. 안드로이드 운영체제에서 영상 정보를 수집하기 위해 사용되는 앱의 경우는 그림 1과 같이 JAVA로 작성된 코드에서 안드로이드 Application Programming Interface(API)를 호출하면 Core Libraries에서 녹화 관련 API가 호출된다. 해

[1] Department of Computer Science, Kyonggi University, Suwon, 443-760, Korea.
[2] Department of Big Data, Pusan National University, Pusan, 609-735, Korea.
[3] School of Computer Science and Engineering, Pusan National University, Pusan, 609-735, Korea.
[*] Corresponding author (yhchoi@pusan.ac.kr)
[Received 18 June 2015, Reviewed 22 June 2015(R2 4 November 2015), Accepted 22 December 2015]
☆ 본 연구는 한국연구재단 논문연구과제(NRF-2013R1A1A1005991) 지원으로 수행되었습니다.
☆ 이 논문은 [21]의 내용을 확장하여 작성되었음.

http://dx.doi.org/10.7472/jksii.2016.17.55



(그림 1) Andorid API를 이용한 화면 정보 수집을 위한 시스템 구조
(Figure 1) System Architecture for Collecting Screen Information Using Andorid API



(그림 2) jni와 fb0를 이용하여 화면 정보를 수집하기 위한 시스템 구조
(Figure 2) System Architecture for Collecting Screen Information Using jni and fb0

당 API는 .class(바이트 코드)로 변환되어 Dalvik VM을 통해 안드로이드 운영체제가 지원하는 Native Libraries를 활용하여 화면 정보를 수집한다. 구체적으로는, 화면 정보를 저장하고 있는 frame buffer(fb)를 접근하기 위하여 Dalvik VM의 java native interface(jni)를 호출하고 디바이스 드라이버인 /dev/graphics/fb0에 접근하여 화면 정보를 수집한다. 이는 jni가 자바 가상 머신이 호스트 운영체제 상의 입/출력과 같은 기능들을 작동하기 위해 로컬 시스템 호출(local system calls)을 수행할 수 있도록 하는 인터페이스를 제공하기 때문에 가능하다.

하지만 그림 1과 같이 안드로이드 API를 이용하여 화면 정보를 수집하는 경우에는 여러 단계(JAVA->Core Libraries->Dalvik VM->Native Libraries 와 Native Services-> Linux Kernel)를 거쳐 화면 정보를 수집하게 되어 7~8 fps(frame per second)의 속도로 정보를 수집하고 이로 인해 화질 저하를 야기한다. 이런 이유로 매끄러운 녹화 및 스트리밍 서비스를 지원하기 위해서는 최적의 화질을 얻기 위한 효율적인 앱 구현 기법이 요구된다.

매끄러운 멀티미디어 스트리밍 서비스를 지원하기 위해서는, 효율적인 비트율 제어 기법 등을 통한 네트워크 관점의 빠른 전송 속도뿐만 아니라 모바일 장치에서 제공되는 서비스의 품질, 즉 음성 및 영상 녹화 속도의 향상 기법이 필요하다[6]-[12]. 그러나 기존 대부분의 연구들은 네트워크 전송 시 스트리밍 서비스의 품질 향상을 위한 데이터 전송 방법의 안정성을 위한 멀티미디어 음성과 영상의 부호화 방법에 대한 연구가 주를 이루었다. 대표적으로는 정지 영상의 경우 최적 비트율 제어를 위해 비터비(Viterbi) 알고리즘을 적용한 방법[13]과 동영상을 위한 프레임간의 의존성을 이용하는 방법[14]이 제안되었다. 또한, 인간의 시각적인 특성을 고려하여 매크로블록을 분류하는 방법[15]과 비트율-왜곡 모델을 이용한 방법[16]이 제안되었다.

본 논문에서는 앱 개발 시점에서의 비효율적인 구현 방식으로 인해 전송할 영상을 저장하는 시점에서의 품질을 보장하지 못한다는 한계를 극복하기 위하여, 그림 2와 같이 jni를 이용하여 fb에 접근하여 화면 정보를 수집한다. 즉, 안드로이드 API를 사용하는 경우와 비교해 Dalvik jni를 활용하여 직접 fb에 정장된 영상을 수집하는 방법으로 평균 23~24fps의 속도로 화면 정보를 수집(JAVA->jni->C/C++ 원시 코드-> Linux Kernel내 fb)한다. 즉, jni를 활용하여 frame buffer에 접근 방식을 통해 화면 정보를 수집하여 매끄러운 영상 녹화 및 스트리밍 서비스가 가능한 새로운 안드로이드 앱인 Mobile Screen Video Recorder using Frame Buffer(FB-MSVR)를 제안한다. MSVR-FB는 매끄러운 영상 녹화 기능과 스트리밍 기능을 지원하기 위해 다음과 같은 특징을 가진다.

- **빠른 녹화 기능**: 안드로이드 하위 버전부터 내장으로 제공하였던 H.264 또는 MPEG-4코덱을 사용하여 안드로이드 버전과 코덱에 종속적이지 않으며, Android fb에 jni를 이용하여 안드로이드 디바이스 화면을 녹화함으로써 녹화 속도를 향상시킬 수 있다.

(표 1) 안드로이드 기본 제공 코덱 요약 [17]
(Table 1) Summary of Android Default Codecs [17]

Type	Format/Codec	Encoder	Decoder
Audio	AAC LC	o	o
	HE-AACv1	o (ver. 4.1+)	o
	HE-AACv2		o
	AAC ELD	o (ver. 4.1+)	o (ver. 4.1+)
	AMR-NB	o	o
	AMR-WB	o	o
	FLAC		o (ver. 3.1+)
	MP3		o
	MIDI		o
	Vorbis		o
	PCM/WAVE	o (ver. 4.1+)	o
Image	JPEG	o	o
	GIF		o
	PNG	o	o
	BMP		o
	WebP	o (ver. 4.0+)	o (ver. 4.0+)
Video	H.263	o	o
	H.264 AVC	o (ver. 3.0+)	o
	MPEG-4 SP		o
	VP8	o (ver. 4.3+)	o (ver. 4.3+)
Flash	adobe flash 10.1지원 (ver. 2.2 +)		

- **이식성이 뛰어난 미디어 파일 포맷 지원**: 안드로이드 디바이스의 제한적인 무선 데이터 사용량과 안드로이드 운영체제와 웹에서 사용 가능하고, 안드로이드 내장 코덱중 하위 버전부터 지원했던 flash video(flv)를 이용하여 미디어 파일 포맷인 플래시 비디오(flash video) 파일을 이용하여 스트리밍 서비스를 제공할 수 있다.

본 논문은 다음과 같이 구성하였다. 먼저 단락 2의 관련 연구에서는 기존 화면 녹화 프로그램의 특성을 분석한 결과를 기술하고, 단락 3에서는 제안한 앱의 구성 및 각 기능 별 동작 시퀀스를 기술하였다. 단락 4에서는 실제 앱의 구현 후 동작 화면을 기술하고, 단락 5에서는 성능 분석 결과를 기술하였다. 마지막으로 단락 6에서 본 논문을 요약하였다.

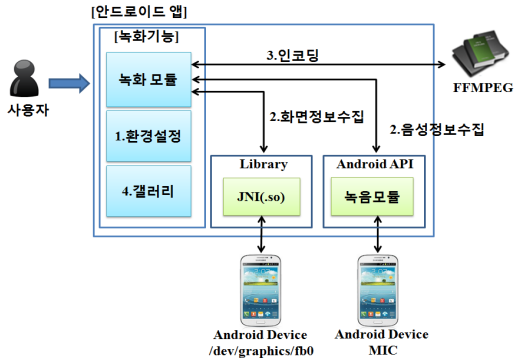
2. 관련 연구

표1에서 안드로이드에서 기본 제공하는 코덱을 요약하였다[17]. 안드로이드에서는 Third-Party와 Hareware Media Codec, I/O 장치와 콘텐츠 정책을 위한 인터페이스를 지원하며 3GPP, MPEG-4, AAC, MP3, MPEG-4, H.263, AVC, JPEG 등의 비디오와 이미지를 위한 인코더/디코더, AMR-NB, AMR-WB를 포함한 스피치 코덱, MP3, AAC, AAC+를 포함한 오디오 코덱, 3GPP, MPEG-4, JPEG를 포함한 미디어 레코딩, 324-M 표준 기반의 화상 통화, adobe flash등을 지원한다. 안드로이드의 상위 버전이 출시되면서 지원 가능 한 코덱의 종류가 추가되고 있다. 또한 스마트폰 제조사 별로 추가 코덱을 제공하여 디바이스 별로 지원 가능한 코덱의 종류가 다르다. 따라서 스마트폰 디바이스와 안드로이드 버전에 제약 받지 않기 위해서는 안드로이드 내장 코덱중 하위 버전부터 지원했던 flash video(flv)를 이용하여야 한다[1]-[5].

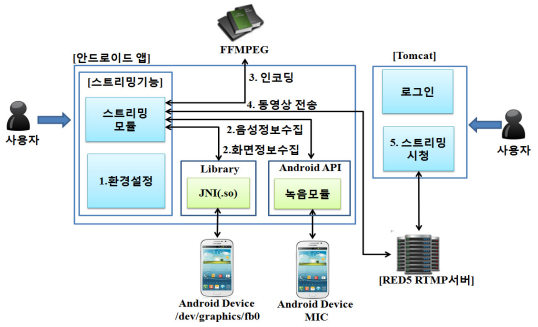
제안한 안드로이드 스마트폰 디바이스의 화면을 녹화하고 스트리밍 서비스하는 앱은 Android 자체 지원 코덱을 이용한 인코딩을 통해 디바이스 제조사별 추가 지원 코덱에 영향을 받지 않고 녹화 파일의 재생이 가능하며, 스트리밍 기능을 통해 다른 사용자와 스마트폰 화면을 공유 할 수 있는 특징을 갖는다. 또한 스트리밍 시 flv를 사용하므로 web을 통해 다른 종류의 디바이스에서도 시청이 가능하다는 장점이 있다. 현재 스트리밍 기능의 경우 한 가지 스트리밍만 이용가능하게 구현이 되어있다.

3. 앱 구성 및 동작

제안한 앱에서는 FFMPEG 오픈 소스 라이브러리[19]를 사용하였다. FFMPEG 오픈 소스 라이브러리는 MPEG, H.263, H.264, ACC 등 371가지의 오디오, 비디오 코덱을 지원하며 영상에 대하여 다양한 형태로의 인코딩, 디코딩을 지원하며 MPlayer프로젝트 서버를 호스팅하고 있다. 따라서 녹화 및 스트리밍 시 인코딩을 하기 위해 FFMPEG 오픈 소스 라이브러리를 사용하였다. 녹화 기능에서는 범용적으로 사용되는 MPEG-4와 ACC코덱을 통해 인코딩을 하며 스트리밍 기능에서는 flv를 이용하여 인코딩하였다. 이는 어도비 시스템즈(Adobe Systems)에서 지원하는 플래시 비디오(flv)는 용량을 줄이면서 화질은 유지할 수 있기 때문에 무선네트워크 전송에 따른 데이터 사용량을 고려할 수 있으며 android 환경뿐만 아니라 web과 스트리밍 환경에서도 효과적으로 사용이 가능하기 때문이다.



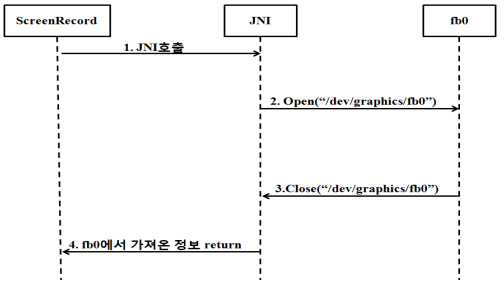
(그림 3) 녹화 기능을 지원하기 위한 이벤트 흐름
(Figure 3) Event Flow Diagram for Video Record



(그림 4) 스트리밍 기능을 지원하기 위한 이벤트 흐름
(Figure 4) Event Flow Diagram for Video Stream

제안한 앱은 기능적으로 /dev/graphics/fb0에서 가져온 화면 정보와 디바이스의 마이크를 이용해 수집한 음성 및 영상 정보를 FFMPEG를 통해 녹화하는 기능과 Red 5 서버[19]를 통한 스트리밍 기능으로 구분된다.

3.1 녹화 기능 동작



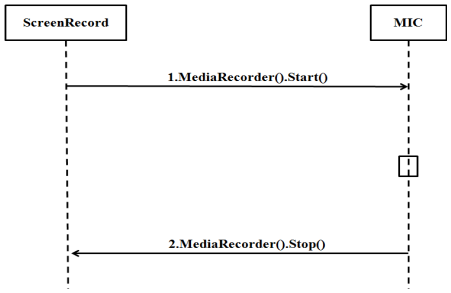
(그림 5) 화면 정보 수집 동작
(Figure 5) Work Flow for Collecting Screen Information

녹화 기능은 스마트폰 디바이스의 현재 진행 중인 화면을 동영상 포맷으로 인코딩하여 SD Card에 저장하는 동작을 수행한다. 녹화 기능은 그림 3에서 요약한 바와 같이 다음의 순서를 따라 동작한다.

1. 환경설정 부분에서 초당 프레임 수(fps), 인코딩 후 녹화 파일 저장 이름을 설정한다.
2. 녹화모듈을 통해 녹화 기능을 수행하게 된다. 녹화 모듈에서는 jni로 작성된 fb0의 정보를 가져오는 라이브러리를 호출한다. 해당 라이브러리에서는 스마트폰 디바이스의 "/dev/graphics/fb0" 디바이스 드라이버를 호출하여 화면 정보를 얻어온다. 또한, Android API의 녹음 모듈을 통해 디바이스의 마이크를 이용하여 음성 정보를 수집한다.
3. 수집한 음성 및 영상 정보를 이용하여 인코딩에서는 FFMPEG 오픈 소스 라이브러리를 통해 MPEG-4을 사용하여 동영상 포맷으로 인코딩하여 환경 설정에서 지정한 이름으로 해당 동영상을 SD Card에 저장한다.
4. 갤러리를 통해 저장된 동영상을 시청한다.

3.2 스트리밍 기능 동작

스트리밍 기능은 스마트폰 디바이스의 현재 진행 중인 화면을 동영상 포맷으로 인코딩하여 Red5 RTMP 서버로 전송하며, Tomcat web서버를 통해서 스트리밍을 시청하는 동작을 수행한다. 스트리밍 기능은 그림 4에서 요약한 바와 같이 다음의 순서를 따라 동작한다.



(그림 6) 음성 정보 수집 동작
(Figure 6) Work Flow for Voice Record

1. 환경설정 부분에서 초당 프레임 수(fps)를 설정한다.
2. 스트리밍 모듈을 통해 스트리밍 기능을 수행한다. 이때, 스트리밍 모듈에서는 녹화 기능에서와 같이 jni와 녹음 모듈을 이용하여 화면과 음성 정보를 수집한다.

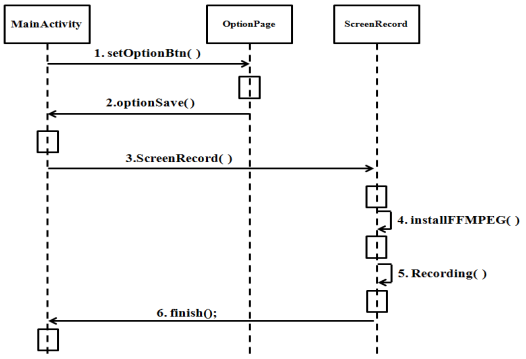
3. 인코딩에서는 FFMPEG 오픈 소스 라이브러리를 통해 flv를 사용하여 동영상 포맷으로 인코딩한다.
4. 스트리밍 서버 전송과정을 통해 Red5 RTMP 서버로 인코딩한 동영상 데이터를 전송합니다.
5. 스트리밍을 시청하고자 하는 사용자는 Flash Player가 설치된 디바이스를 통해 웹서버에 접근하여 스트리밍을 실시간으로 시청할 수 있다.

3.3 상세 작업 플로우

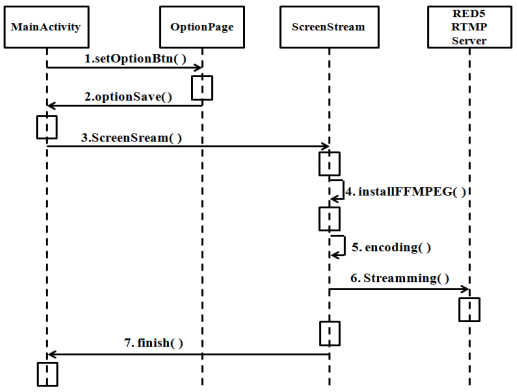
3.3.1 화면 정보 수집 동작 시퀀스

안드로이드 디바이스 상의 Frame Buffer에 접근하여 화면 정보를 수집하기 위해서는 Application계층에서 직접 접근이 불가능하므로 jni를 통해서 접근한다. 그림 5에서 요약한 바와 같이 화면 정보를 수집하기 위해 다음의 동작을 수행한다.

1. JAVA로 작성된 안드로이드 소스에서 jni를 호출한다. FrameBuffer의 종류는 fb0과 fb1이 있다. fb0에는 현재 표현하고 있는 화면의 정보를 가지고 있으며 fb1에는 다음에 표현할 화면의 정보를 가지고 있다. 따라서 jni에서는 fb0에서 화면 정보를 얻어 온다. jni로 작성된 파일은 .so파일로 프로젝트 내에 위치합니다.
2. 호출된 jni는 FrameBuffer의 디바이스 드라이버인 /dev/graphics/fb0에 접근한다. 이 때, Open("dev/graphics/fb0")을 통해 fb0에 접근하여 화면정보를 받아 온다.
3. Close("/dev/graphics/fb0")을 통해 열었던 디바이스 드라이버를 닫는다.
4. fb0에서 가져온 정보의 return 과정을 통해 화면 정보를 수집한다.



(그림 7) 녹화 기능 동작
(Figure 7) Work Flow for Screen Record



(그림 8) 스트리밍 기능 동작
(Figure 8) Work Flow for Video Stream

3.3.2 음성 정보 수집 동작 시퀀스

음성 정보의 수집은 Android에서 제공하는 API를 이용하여 그림 6에서 요약한 바와 같이 다음과 같이 한다.

1. MediaRecorder().Start()를 호출하여 디바이스 상의 마이크를 이용하여 음성 정보를 수집한다.
2. MediaRecorder().Stop()을 호출하여 음성 정보 수집을 종료한다.

3.3.3 녹화 기능 동작 시퀀스

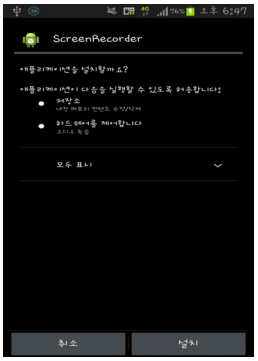
녹화 기능의 경우 그림 7에서 요약한 바와 같이 다음과 같이 영상을 녹화한다.

1. setOptionBtn()을 호출하여 OptionPage로 이동하여 녹화를 위한 환경 설정을 한다.
2. optionSave()를 호출하여 환경 설정 변경 값을 저장한다.
3. ScreenRecord()를 호출하여 해당 앱은 서비스계층의 백그라운드에서 동작한다.
4. installFFMPEG()를 호출하여 FFMPEG오픈 소스 라이브러리를 설치한다.
5. Recording()을 호출하여 jni 호출을 통한 fb0에서의 화면 정보를 수집하고, Android API을 통해 디바이스 마이크를 이용한 음성 정보 수집을 수행한다. 이렇게 수집한 정보를 가지고 FFMPEG 오픈 소스 라이브러리를 통해 MPEG-4 동영상 포맷으로 인코딩하여 디바이스의 SD Card에 저장한다.
6. finish()가 호출되면 백그라운드에서 동작하던 서비스가 종료되며 녹화를 마치고 메인화면으로 돌아온다.

3.3.4 스트리밍 기능 동작

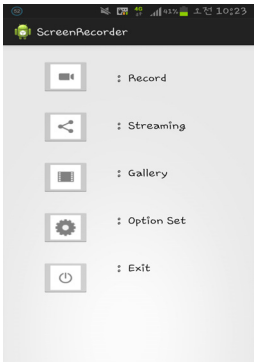
스트리밍 기능은 그림 8에서 요약한 바와 같이 다음의 동작 순서를 따른다.

1. setOptionBtn()에서 installFFMPEG() 호출까지 녹화 기능과 동일한 동작을 수행한다. (그림 8의 1~4 단계)
2. encoding()에서 jni호출을 통해 fb0에서의 화면정보를 수집하고, Android API을 통한 디바이스 마이크를 이용한 음성 정보를 수집하여 수집한 정보를 가지고 FFMPEG 오픈 소스 라이브러리를 통해 flv 동영상 포맷으로 인코딩 한다. (그림 8의 5 단계)
3. Streaming()을 호출하여 무선 네트워크를 통해 동영상 스트림을 Red 5 RTMP 서버로 전송한다. (그림 8의 6 단계)
4. finish()가 호출되면 백그라운드에서 동작하던 서비스가 종료되며 스트리밍을 마치고 메인 화면으로 복귀한다. (그림 8의 7 단계)

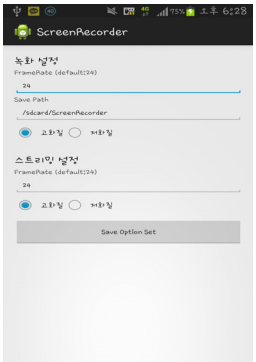


(a)설치화면

(b)루트요청 화면



(c)메인화면



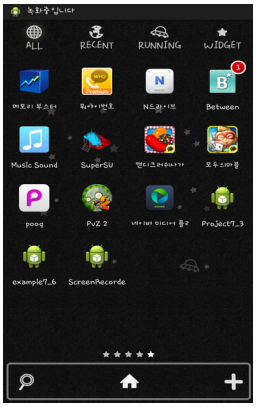
(d)환경설정화면

(그림 9) 개발한 앱의 설치, 실행(루트 요청과 메인 화면) 및 환경 설정 화면

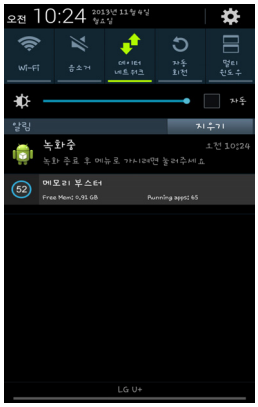
(Figure 9) App Installation, Execution and Setup

4. 그래픽 사용자 인터페이스

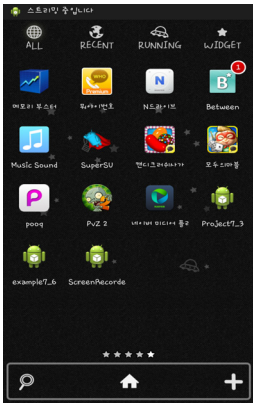
그림 9에서 개발된 앱의 설치(그림 9(a)), 루트권한 요청(그림 9(b)), 메인(그림 9(c)) 및 환경설정(그림 9(d)) 화면을 각각 나타내었다. 어플리케이션을 설치한 후 실행하면 최초 1회 루트 권한을 요청한다.*주1) 메인 화면에서 Option Set 버튼을 누르면 환경 설정 페이지로 이동하며 이곳에서는 녹화의 경우 저장파일의 이름과 화질(고화질 혹은 저화질)을 설정할 수 있다. 설정 후 Save Option Set 버튼을 누르면 메인화면으로 돌아온다.



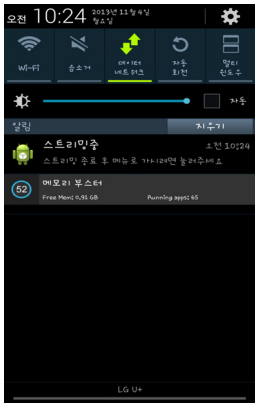
(a)녹화 중 화면



(b)녹화 중 상태 표시



(c)스트리밍 진행 중 화면

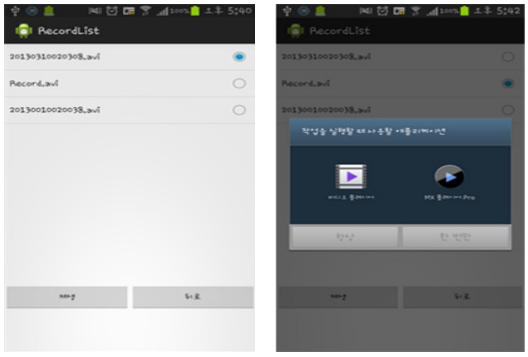


(d)스트리밍 진행 중 상태 표시

(그림 10) 개발한 앱에서의 녹화, 스트리밍 실행 화면

(Figure 10) Video Recording and Streaming

*주1) 스마트폰에서 루트 권한을 획득하는 과정에서 보안 측면이 약화될 수 있음. 안드로이드 운영 체제 보안을 통해 루팅으로 인한 보안 문제를 해결하기 위한 노력이 필요.



(a)녹화파일 조회 (b)파일 선택 재생

(그림 11) 개발한 앱에서의 녹화파일 조회 및 재생 화면

(Figure 11) File Search and Play

(a)웹서버 로그인 (b)스트리밍 시청

(그림 12) 스트리밍 시청 웹 서버 화면 화면

(Figure 12) Web Server Screen for Streaming Service

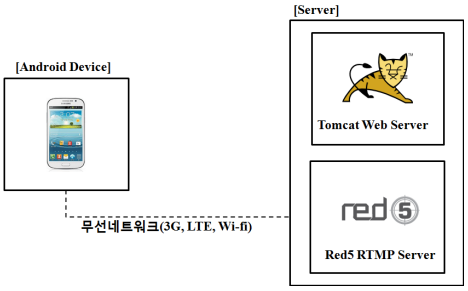
그림 10에서 개발된 앱의 녹화 중 화면(그림 10(a)), 녹화 중 상태 표시(그림 10(b)), 스트리밍 중(그림 10(c)) 화면, 스트리밍 진행 중 상태 표시(그림 10(d)) 화면을 나타내었다. 녹화 또는 스트리밍 기능을 실행하면 어플리케이션이 서비스계층에서 백그라운드로 실행되며 화면 상단의 상태 바에 '녹화중입니다' 또는 '스트리밍중입니다'라는 알림 메시지를 띄운다. 녹화 또는 스트리밍을 마치고자 할 때에는 상태 바를 내려 '녹화 중' 또는 '스트리밍 중' 상태 표시(notification)를 클릭하면 녹화 또는 스트리밍을 종료한 후 메인화면으로 화면이 복귀한다.

그림 11은 녹화 파일 조회(그림 11(a)) 화면과 파일 선택 재생(그림 11(b)) 화면을 나타낸다. 메인 화면에서 겔러리 버튼을 누르면 녹화 파일 조회 화면으로 이동하며 해당 어플리케이션에서 녹화한 파일의 리스트가 출력된다. 녹화한 파일 중 하나의 파일을 선택하고 재생 버튼을 누르면 그림 11(b) 화면과 같이 디바이스에 설치된 동영상 플레이어 중 재생하고자 하는 플레이어를 선택하는 화면이 나오며 선택한 플레이어를 통해 재생한다.

그림 12는 웹서버 로그인(그림 12(a))과 스트리밍 시청(그림 12(b)) 화면을 보여준다. 스트리밍 시청 웹페이지에 접속하면 로그인 화면이 나타나며 로그인 후에는 Red 5 Server에서 제공하는 스트리밍 페이지로 이동된다. 해당 페이지를 통해 스트리밍을 시청할 수 있다.

5. 성능 실험

5.1 실험 구성



(그림 13) 실험 환경 구성

(Figure 13) Experimental Environment

그림 13과 같이 실험 환경을 구성하고 기존 API와 jni를 사용한 화면 정보를 획득하는 성능을 비교하기 위한 실험을 하였다. 이를 위해, 다음과 같이 안드로이드 스마트 폰과 서버 환경을 구성하였다.

o 안드로이드 스마트폰 환경
- 테스트 안드로이드 기기: 갤럭시 노트 2
- 개발프로그램: eclipse juno 64bit
- 안드로이드 버전: 젤리빈 4.1.2
- 안드로이드 NDK
  android-ndk-r9d-windows-x86_64bit

o 서버 환경
- Server OS: Windows 7 64bit Ultimate
- WebServer: Tomcat 7.0
- Streaming Server: Red5 1.0.1
- 코덱 정보 확인(영상길이, 코덱종류확인): mediainfo [20]

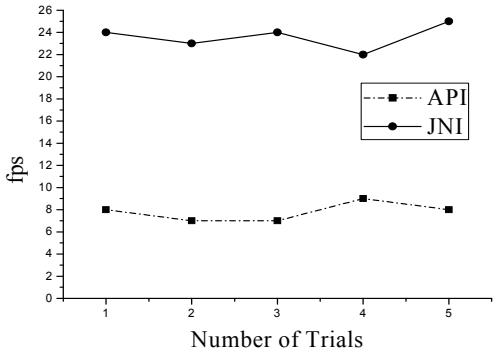
(표 2) 기존 음성 및 화면 녹화 앱과 제안한 앱(FB-MSVR)의 성능 비교 분석
(Table 2) Performance Comparison of Current Apps and the Proposed App(FB-MSVR) for Recording Voice and Screen

이름	녹화			스트리밍			기타
	화질(fps)	코덱	메모리	화질(fps)	코덱	메모리	
Screen Cast video recorder [1]	정상작동 안됨		8.9MB	지원하지 않음			-
Screen video recorder [2]	24	MPEG-4 오디오 지원 X	39MB	지원하지 않음			영상정보 기록 지원 X
Z-screenrecorder [3]	정상작동 안됨		28MB	지원하지 않음			-
AfreecaTV [4]	데스크톱과 연결해서 동작				H.264		-
mobizen [5]	데스크톱과 연결해서 동작			스트리밍 지원하지 않음			-
FB-MSVR (Proposed)	**24**	**MPEG-4 AC-3**	**5.2MB**	**24**	**flv**	**5.4MB**	**-**

5.2 기존 화면 녹화 프로그램과의 비교 실험

표 2에서 현재 사용 중인 상용 안드로이드 스마트폰 녹화 앱의 성능과 제안한 앱의 성능을 실험한 결과를 요약하였다.

기존 프로그램 중 Screen video recorder는 MPEG-4를 이용하여 녹화는 정상적으로 되었으나 녹화 프로세스가 관리되지 않아 녹화된 파일의 길이 정보, 초당 프레임 수 정보를 볼 수 없으며 오디오를 지원하지 않고 스트리밍 서비스도 제공하지 않는 특성을 갖는다. 또한, Screen Cast video recorder, Z-screenrecorder 의 경우에도 특정기기에서 제한적으로 동작하는 문제가 있기 때문에 녹화가 정상적으로 진행되지 않았으며 프로세스 관리가 되지 않아 메모리 누수가 발생하는 등 여러 가지 문제점을 발생하였다.



(그림 14) 각 실험별 API와 jni를 사용하는 각각의 평균 fps
(Figure 14) Average fps for API and jni

(표 3) 각 시도 횟수 별 녹화된 fame 수
(Table 3) Number of Record Frames

시도 횟수	1	2	3	4	5	평균 (fps)
API	81	74	73	89	82	7.8
jni	243	234	242	229	248	23.6

또한 스트리밍 서비스를 제공하지 않는 단점을 가지고 있었다. 현재 가장 보편적으로 사용되고 있는 AfreecaTV의 경우에는 안드로이드 디바이스 화면의 녹화 및 스트리밍 기능은 별도의 데스크톱과 연결해서 동작하며 안드로이드 디바이스만을 이용한 스트리밍 기능은 카메라 모듈을 이용하여 수집한 영상 정보를 H.264를 이용하여 스트리밍 서비스를 제공하고 있다. mobizen의 경우도 별도의 데스크톱과 연결해서 동작하며 데스크톱 내에서 녹화를 진행한다.

5.3 FB-MSVR의 평균 녹화된 fps

실험은 총 5회씩 수행하였으며 1회에 10초 동안 측정한 프레임 수를 1초당 얻어오는 프레임 수를 평균하여 수치화 하였다. 그림 14에서 화면 정보를 얻어오는 Android API와 NDK를 이용한 jni를 통해 Native영역의 fb0에 접근하여 화면 정보를 얻어오는 성능을 측정한 실험 결과를 요약하였다. 표3에서는 각 시도 회수 별로 녹화된 frame 수를 측정하여 기록하였다.

실험 결과 Android API는 5회 실험 평균 초당 7.8프레임, jni(FB-MSVR)를 통해서는 초당 23.6프레임 이라는 결

과를 얻을 수 있었다. 따라서 Android API를 통해 화면 정보를 얻어올 경우 초당 프레임수를 충족시키지 못하여 인코딩한 결과가 매끄럽지 못한 영상을 나타내는 반면에, jni를 통해 fb0에서 화면 정보를 얻어 오는 경우에는 초당 20프레임 이상의 수를 충족시키고 있다. 일반적으로 화면이 부드럽게 재생이 되기 위해서는 초당 15fps가 되어야 하며 색상수가 많은 실사의 경우 최소 20fps, 실사에 비해 색상수가 적은 애니메이션의 경우 최소 15fps을 필요로 하는 점을 고려하면, 이 실험 결과는 제안한 FB-MSVR이 매끄러운 영상을 나타낼 수 있음을 나타낸다.

6. 결 론

본 논문에서는 녹화된 화면 정보를 미디어 파일 포맷인 플래시 비디오(flash video) 파일을 이용하여 실시간으로 스트리밍 서비스를 제공하기 위한 새로운 앱을 제안하였다. 이는 기존의 안드로이드 화면 녹화 앱이 가지고 있는 녹화 기능의 코덱, 안드로이드 운영 체제 버전에 종속적인 특성과 별도의 데스크톱 PC와 유선으로 연결하여 서비스를 제공해야 하는 단점을 극복함으로써, 부호화 방법의 변경 없이 초당 20프레임 이상의 화면 녹화를 통해 매끄러운 영상을 녹화하는 것이 가능함을 실험적으로 확인하였다. 이를 통해 개발된 앱은 안드로이드 스마트폰 사용자의 실시간 스트리밍 화면 시청 시에 효율적으로 활용될 수 있을 것으로 기대된다.

참 고 문 헌 (Reference)

[1] "Screen Cast video recorder," https://play.google.com/store/apps/details?id=com.ms.screencastfree

[2] "Screen video recorder v3.0," https://play.google.com/store/apps/details?id=com.cocoapps.screenrecorder

[3] "Z-screenrecorder," https://play.google.com/store/apps/details?id=com.zausan.zscreenrecorder

[4] "AfreecaTV," https://play.google.com/store/apps/details?id=kr.co.nowcom.mobile.afreeca

[5] "Mobizen," http://www.mobizen.com/

[6] H. Kim, J. Koo, and K. Chung, "Equation-based Quality Control Scheme for Improving QoE of Multimedia Streaming Service," Proc. of the KIISE Fall Conference, vol.37, no.2(B), pp.131-135, Nov. 2010. (in Korean) http://www.dbpia.co.kr/Article/NODE01547890

[7] J. Koo and K. Chung, "A Novel Rate Control for Improving the QoE of Multimedia Streaming Service in the Internet Congestion," Journal of KIISE: Information Networking, vol.36, no.6, pp.492-504, Dec. 2009. (in Korean) http://www.dbpia.co.kr/Article/NODE01311128

[8] B. Wang, W. Wei, Z. Guo, and D. Towsley, "Multipath Live Streaming via TCP: Scheme, Performance and Benefits," Proc. of the ACM CoNEXT, Dec. 2007. http://dx.doi.org/10.1145/1364654.1364668

[9] D. Nguyen and J. Ostermann, "Congestion Control for Scalable Video Streaming Using the Scalability Extension of H.264/AVC," IEEE Journal of Selected Topics in Signal Processing, vol.1, no.2, pp.246-253, Aug. 2007. http://dx.doi.org/10.1109/JSTSP.2007.902068

[10] B. Libaek and O. Kure, "Generic Application Level Rate Control for Scalable Video Using Shadow Probing," Proc. of the ICSNC, pp.131-137, Sep. 2009. http://dx.doi.org/10.1109/ICSNC.2009.85

[11] R. Rejaie, M. Handley, and D. Estrin, "Layered Quality Adaptation for Internet Video Streaming," IEEE Journal of Selected Areas in Communications, vol.18, no.12, pp.2530-2543, Dec. 2000. http://dx.doi.org/10.1109/49.898735

[12] M. Zink, J. Schmitte, and R. Steinmetz, "Layer-Encoded Video in Scalable Adaptive Streaming," IEEE Transaction on Multimedia, vol.7, no.1, pp.75-84, Feb. 2005. http://dx.doi.org/10.1109/TMM.2004.840595

[13] A. Ortega, K. Ramchandran, and M. Vetterli, "Optimal trellis-based buffered compression and fast approximations," IEEE Tans. Image Processing, vol. 3, no. 1, pp. 26-40, Jan. 1994. http://dx.doi.org/10.1109/83.265978

[14] K. Ramchandran, A. Ortega, and M. Vetterli, "Bit allocation for dependent quantization with applications to multiresolution and MPEG video coder," IEEE Tans. Image Processing, vol. 3, no. 5, pp. 533-545, Sep. 1994. http://dx.doi.org/10.1109/83.334987

[15] ISO-IEC/JTC1/SC29/WG11, “Test Model 5,” Draft, April 1993.
http://www.mpeg.org/MPEG/MSSG/tm5/Overview.html
[16] J. Zdepsky, D.Raychaudhuri, and K. Joseph, “Statistically based buffer control policies for constant rate transmission of compressed digital video,” IEEE Tans. Commun., vol. 39, no. 6, pp. 947-957, June 1991.
http://dx.doi.org/10.1109/26.87184
[17] ″Embedded Android Codec, ″
http://developer.android.com/guide/appendix/media-formats.html
[18] ″FFMPEG, ″ http://ffmpeg.mplayerhq.hu/
[19] “Red 5,” http://www.red5.org/
[20] “Mediainfo,” http://mediaarea.net/ko/MediaInfo
[21] S.-M. Seo and Y.-H. Choi, “Development of Android App for Smooth Multimedia Streaming Service via Portable Media File Format”, International Journal of Multimedia and Ubiquitous Engineering, vol. 9, no. 10, pp. 197-208, 2014.
http://dx.doi.org/10.14257/ijmue.2014.9.10.19

◐ 저 자 소 개 ◑

서 상 민 (Sang-min Seo)
2008년 경기도 용인 구성고등학교
2013년 경기대학교 컴퓨터과학과(학사)
관심분야 : JAVA, Android, Embedded System, etc.
E-mail : sang-min@kyonggi.ac.kr

권 준 호 (Joonho Kwon)
1999년 서울대학교 컴퓨터공학과 졸업(학사)
2001년 서울대학교 대학원 전기컴퓨터공부 졸업(석사)
2009년 서울대학교 대학원 전기컴퓨터공학부 졸업(박사)
2010～현재 부산대학교 빅데이터협동과정 교수
관심분야 : 데이터사이언스. 빅 데이터 처리 및 분석, IoT 데이터 저장 및 관리, 그래프 데이터베이스
E-mail : jhkwon@pusan.ac.kr

최 윤 호 (Yoon-Ho Choi)
2004년 서울대학교 전기컴퓨터공학과(공학석사)
2008년 서울대학교 전기컴퓨터공학과(공학박사)
2009년 펜실베니아주립대학교 컴퓨터과학과(박사후연구원)
2010년～2011년 삼성전자 DMC연구소 & 네트워크사업부(책임 연구원)
2012년～2014년 경기대학 융합보안학과 교수
2014년～현재 부산대학교 정보컴퓨터공학부 교수
관심분야 : 침입 탐지, SDN/NFV 보안, 웜/바이러스 차단, 모바일 앱 보안, 지능형 차량 보안, etc.
E-mail : yhchoi@pusan.ac.kr