안드로이드 환경에서 단말 대 단말 실시간 미디어 전송 기법

임효영*, 박영민**, 이상아***, 곽종욱*
*영남대학교 컴퓨터공학과
**영남대학교 전기공학과
***경북대학교 IT대학 컴퓨터학부
e-mail : hyoyoung_@naver.com

End-to-end Real Time Media Transport Technique in Android Platform

Hyo Young Lim*, Young Min Park**, Sang Ah Lee***, Jong Wook Kwak*
*Dept. of Computer Engineering, Yeungnam University
**Dept. of Electrical Engineering, Yeungnam University
***School of Computer Science and Engineering,
Kyungpook National University

요 약

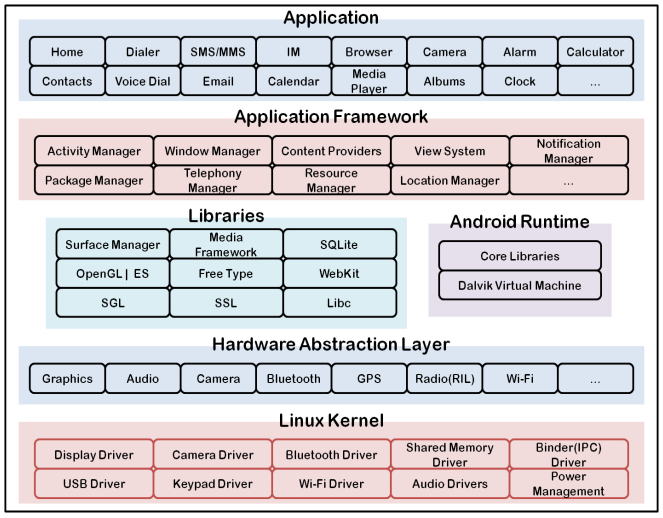
현재 방송국이 제작하는 방송이 아닌 개인이 제작하는 방송은 인터넷 환경에서 생겨났고 발전해왔다. 이에 따라 개인 역시 불특정 다수에게 시간적 제약 없이 방송을 할 수 있다. 하지만 여전히 인터넷이 지원되는 컴퓨터 단말 앞에서라는 공간적 제약이 존재한다. 본 논문에서는 이러한 공간적 제약을 극복하고자 장소에 구애받지 않는 모바일 환경에서 단말 대 단말 실시간 이미지 전송 기법을 제안하여 실시간 방송을 가능하게 하였다. 안드로이드 환경에서 제안한 본 기법은 무선 인터넷(Wi-Fi)이 지원되는 환경이라면 어디든 가능하다는 점에서 공간적 제약이 없어져 효율적이다. 성능평가 결과 안정적인 무선 인터넷이 지원되는 환경에서 지연시간 5000ms이상일 때 100프레임 당 10번 이하의 깜박임 현상(끊김 현상)이 나타나 안정적으로 동작 하는 것을 확인하였다.

1. 서론

최근 *Twitter*와 *me2Day*같은 소셜 네트워크 서비스(SNS)가 인기를 얻고 있다. 이는 빠른 전파 속도와 쉬운 접근성과 같은 소셜 네트워크 서비스 고유의 특징에 기인한 것이다. 이에 따라 소셜 네트워크 서비스가 새로운 사회 관계 형성에 큰 영향을 주면서 개인 홍보의 필요성도 급증하게 되는데 그 결과 UCC제작, 개인의 실시간 방송 등이 점차 늘어나는 추세이다[1].

하지만 현재 개인이 실시간 방송을 하기 위해서는 인터넷이 지원되는 컴퓨터 단말 환경에서만 할 수 있다는 공간적 제약이 존재한다. 이러한 제약을 극복하기 위해 본 논문에서는 공간에 구애받지 않는 모바일 환경과 무선 인터넷(Wi-Fi)을 이용하여 안드로이드 환경에서 단말 대 단말 실시간 미디어 전송 기법을 제안하여 실시간 방송을 가능하게 하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문의 배경지식을 알아보고, 3장에서는 전체 시스템 구성과 세부 동작에 대하여 설명한다. 4장에서는 개발 및 성능평가 환경을 소개하고 본 논문에서 제안한 기법의 성능에 대해 분석하고 토의한다. 마지막으로 5장에서는 본 시스템에 대한 결론에 대해서 논의한다.



(그림 1) 안드로이드 운영체제의 주요 구성 요소

2. 배경지식

2.1 안드로이드 플랫폼

안드로이드는 운영체제와 미들웨어 그리고 핵심 어플리케이션을 포함하고 있는 모바일 디바이스를 위한 소프트웨어 스택이며 플랫폼이다.

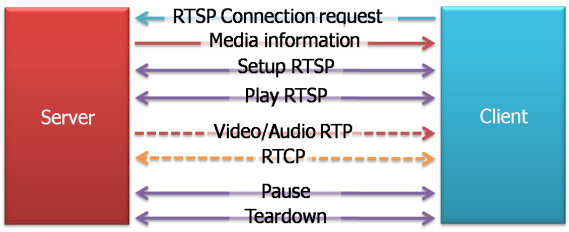
(그림 2) 오픈 핸드셋 얼라이언스

그림 1은 안드로이드 운영체제의 주요 구성 요소를 보여주는 다이어그램이다[2]. 안드로이드는 리눅스 커널[1] 위에서 동작하며, 다양한 안드로이드 시스템 컴포넌트에서 사용되는 C/C++ 라이브러리들을 포함하고 있다. 또한 안드로이드는 기존의 자바 가상 머신과는 다른 달빅 가상 머신을 통해 자바로 작성된 어플리케이션을 별도의 프로세스에서 실행하는 구조로 되어있다.

안드로이드는 구글(Google)의 제품으로 알려져 있지만, 사실 그림 2에 나타나 있는 오픈 핸드셋 얼라이언스(Open Handset Alliance)[2]에서 주도하는 제품이라 보는 것이 더욱 적절하다. 현재 OHA에서 안드로이드 공개 표준을 계속 개발하고 있다. 또한 안드로이드 플랫폼 기반의 스마트폰은 경쟁적으로 개발되는 있는 추세이며, 국내외 유명 단말기 제조사들이 앞 다투어 생산하고 있다[3].

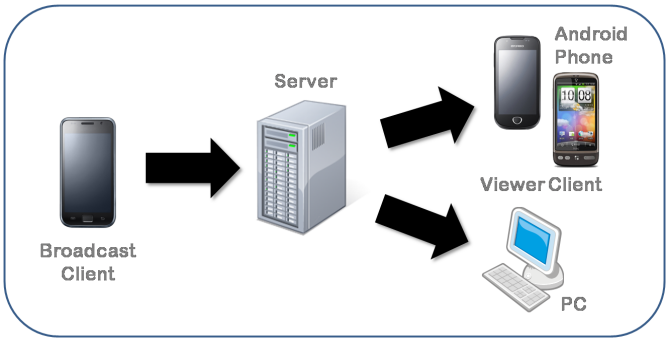
2.2 실시간 방송

실시간 방송은 주로 RTP와 RTSP로 구현되는데, RTP(Real-time Transport Protocol)는 오디오/비디오와 같은 실시간 데이터를 멀티캐스트 혹은 유니캐스트 네트워크를 통해 전송하는데 적합한 단말 대 단말(end-to-end) 전송 프로토콜이다. RTP는 자원 예약을 준비하지 않으며, 실시간 서비스에 대한 Qos를 보장하지 않는다[4].



(그림 3) RTSP 스트리밍 흐름

1) 안드로이드 2.1 버전의 리눅스 커널은 2.6.29버전이다.
2) Open Handset Alliance(OHA), 현재 약 60여개의 하드웨어, 소프트웨어, 통신 업체가 “더 좋은”, “개방된” 모바일 플랫폼을 마켓에 제공하기 위해 구성한 단체이다.



(그림 4) 시스템 구조

한편 RTSP(Real Time Streaming Protocol)는 실시간 스트리밍 프로토콜이다[5]. RTSP는 스트리밍 시스템에 사용되며, 미디어 서버를 원격으로 제어할 때 쓰인다. 또한 시간 정보를 바탕으로 서버에 접근한다. 그림 3은 RTSP의 스트리밍 흐름을 보여준다[6].

3. 안드로이드 환경에서의 실시간 방송

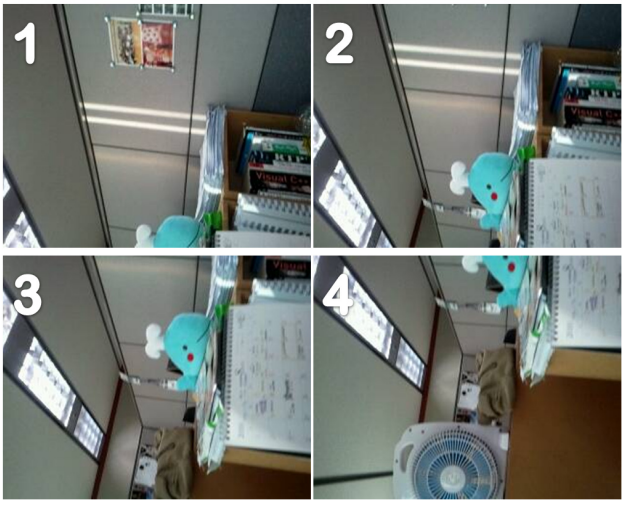
3.1 시스템 구조

그림 4는 본 논문의 시스템 구조를 보여준다. 먼저 방송 클라이언트(Broadcast Client)는 송신매체에 해당하는 클라이언트로 직접 방송을 하기 위해 실시간으로 서버에 실제 방송 데이터와 방송 정보를 전송한다. 그 다음 시청 클라이언트(Viewer Client)는 수신매체에 해당하는 클라이언트로 서버에 방송 데이터를 요청하여 방송을 보거나 다른 기능을 수행하는 클라이언트다. 마지막으로 서버(Server)는 방송 클라이언트가 전송해준 데이터를 시청 클라이언트가 요청하였을 때 전송해주는 데이터 중개자 역할을 수행한다.

3.2 각 모듈 별 구현 및 동작

3.2.1 방송 클라이언트(Broadcast Client)

방송 클라이언트는 직접 방송하는 클라이언트로 서버에게 방송 이름과 태그 같은 방송 정보 및 영상과 음성 같은 실제 방송 데이터를 전달한다.



(그림 5) 방송 클라이언트에 저장된 JPEG 파일들

생방송처럼 직접 찍어 실시간으로 방송하는 라이브 방송을 할 때, 먼저 영상은 Camera.PreviewCallback[3] 의 *void onPreviewFrame (byte[] data, Camera camera)*에 있는 *data*를 사용한다. YUV 색공간(비율은 4:2:0)으로 되어있는 *data*를 실시간으로 받아와서 RGB565의 Bitmap[4]으로 변환하고, 그 변환된 Bitmap을 다시 그림 5에 보이는 바와 같이 JPEG 파일로 압축한 다음 서버로 전송한다. 그리고 음성은 AudioRecord[5]의 *int read (byte[] audioData, int offsetInBytes, int sizeInBytes)*로 읽은 *audioData*를 실시간으로 서버에 전송한다.

3.2.2 시청 클라이언트(Viewer Client)

시청 클라이언트는 방송을 시청하는 기능과 방송을 시청하지 않는 기능으로 크게 두 가지로 나누어 구현한다.

방송을 시청하는 기능은 서버에 특정 방송 클라이언트의 방송 데이터를 요청하면 서버는 시청 클라이언트에게 방송 데이터를 전송한다. 이 때, 영상은 서버에서 받은 JPEG파일을 BitmapDrawable(Bitmap의 Drawble객체)로 변환한 다음 ImageSwitcher[6]에 있는 *void setImageDrawable (Drawable drawable)*을 사용하여 재생한다. 그리고 음성은 서버에서 받은 ByteBuffer객체를 byte배열로 변환한 다음 AudioTrack[7]의 *int write(byte[] audioData, int offsetInBytes, int sizeInBytes)*에 있는 *audioData*에 넣어 재생하게 된다. 그리고 시청 클라이언트에서 방송을 시청 할 때, 실제 방송 클라이언트보다 적게는 1초에서 2초 정도, 크게는 10초 정도 지연시간을 주어 구현하였는데, 이는 무선 인터넷 상태에 따라 생기는 끊기는 현상을 줄이고자 함이다.

또한 방송보기에서 방송에 대한 정보 및 의견 공유를 위한 댓글달기 기능과 방송 클라이언트와 파일을 공유하기 위한 파일공유 기능을 추가로 구현하여 편의를 도모하였다.

그리고 방송을 시청하지 않는 기능에서는 신문보기 기능, 방송통계, 방송추천 기능 등이 있다. 첫 번째로 신문보기 기능은 당일 날짜 신문을 포털 사이트의 API를 통해 가져온 후, 그 신문 기사를 볼 수 있는 기능이다. 두 번째로 방송통계는 방송에 대한 통계를 내서 좀 더 시각적으로 방송에 대한 정보를 볼 수 있게 하는 기능이다. 마지막으로 방송추천 기능은 방송을 시청하기 전, 자신이 자주 보는 방송이나 비슷한 방송을 추천함으로써 검색을 하지 않아도 원하는 방송을 볼 수 있게 하는 기능이다.

3.2.3 서버(Server)

스마트폰 단말기는 보통 무선 인터넷 또는 3G망을 사용한다. 본 논문에서는 무선 인터넷을 이용하는데 스마트폰 단말기의 MAC Address는 고정적이지만, IP Address는 유동적이다. 따라서 단말 간 직접적인 클라이언트 대 클라이언트 통신은 불가능하다. 그러므로 서버 대 클라이언트 모델을 사용한 네트워크 프로그래밍이 불가피 하다.

또한 본 시스템의 서버 모델은 TCP 기반으로 구현한다. 보통 방송 같은 실시간성이 있는 데이터는 UDP로 구현하고 있으나, 본 논문에서는 다음과 같은 세가지 이유로 UDP 대신 TCP로 구현한다. 첫 번째, TCP와 UDP 서버를 따로 둬야 하는 것은 비효율적이다. 두 번째, 방송 데이터를 UDP로 쓰는 이유는 속도가 빠르다는 것이 가장 큰 이유인데, 안드로이드에서는 TCP의 속도와 UDP의 속도가 비슷하거나 TCP가 더 빠른 것을 볼 수 있어서 UDP의 장점을 이용할 수가 없다. 세 번째, 공용 네트워크의 경우, 보안상의 이유로 UDP를 라우터 단계에서 차단해놓은 경우를 어렵지 않게 볼 수 있다. 이 경우, UDP를 사용하면 방송이 불가능해진다. 이러한 이유로 본 논문에서 제안한 서버는 TCP로 구현하였다. 하지만 서버를 TCP로 구현함에 따라 feed back 패킷이 생기는 문제가 생긴다. 이러한 문제점은 방송 클라이언트가 서버에 패킷을 넘길 때 시간정보를 주어 받은 패킷의 시간정보가 이전 패킷의 시간정보 보다 앞설 때 그 패킷은 drop 시키는 것으로 해결하였다.

4. 성능평가

4.1 개발 환경

개발환경은 표 1에 나타 낸 것과 같이 서버는 Window XP Service Pack 3와 클라이언트는 Window7를, 언어는 Java SDK 1.6 버전과 Android SDK 2.1 버전을 사용하였다. 그리고 Tool은 Eclipse SDK 3.5.2를 사용하여 본 시스템을 구현하였다.. 대상 하드웨어는 S사의 SHW-M100S와 M사의 XT720을 사용하여 구현하였다.

<표 1> 개발환경

Develop Environment	
OS	Window XP SP3
	Window 7
Language	Java SDK 1.6
	Android SDK 2.1
Target H/W	SHW-M100S (S사)
	M사 XT720 (M사)
Tool	Eclipse SDK 3.5.2

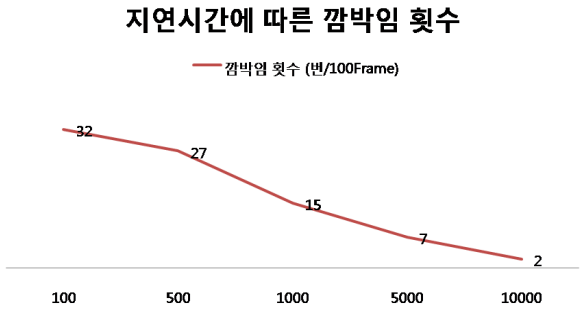
3) Callback 함수. Java의 Listener 와 같은 역할을 하는 함수로써 해당 callback에 해당하는 이벤트가 발생 하였을 때 호출된다. 여기서는 Preview Frame Rate에 맞게 1초당 10번~20번 정도 onPreviewFrame 함수가 호출된다.

4) Bitmap. android.graphics.Bitmap (http://developer.android.com/reference/android/graphics/Bitmap.html 참고)

5) AudioRecord. android.media.AudioRecord (http://developer.android.com/reference/android/media/AudioRecord.html 참고)

6) ImageSwitcher. android.widget.ImageSwitcher (http://developer.android.com/reference/android/widget/ImageSwitcher.html 참고)

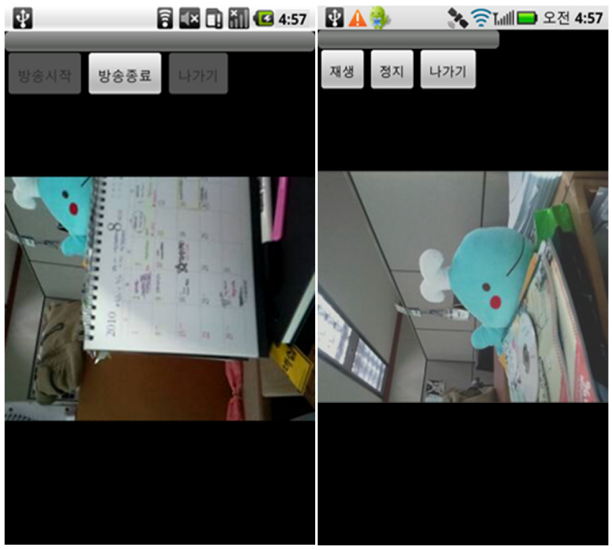
7) AudioTrack - android.media.AudioTrack (http://developer.android.com/reference/android/media/AudioTrack.html 참고)



(그림 6) 지연시간에 따른 깜박임 횟수

4.2 실제 모의 성능 평가 결과

본 논문에서 제안한 실시간 영상 전송 기법을 사용하여 실제 모의 성능 평가 방송을 수행 해 본 결과는 다음과 같다. 무선 인터넷 상태가 안정적인 경우, 표 2에 나타난 것과 같이 지연시간을 약 100ms에서 1000ms정도로 주었을 때는 100프레임 당 각각 약 30회에서 15회 정도의 깜박임 현상(끊김 현상)이 있었으나, 약 5000ms이상의 지연시간을 주었을 때에는 10번 이하의 깜박임 현상이 발생하였다. 이는 지연시간이 증가할수록 깜박임 횟수는 줄어든다는 것을 의미한다. 깜박임 횟수를 줄이기 위해서 5000ms의 지연시간을 준 결과가 그림 6에서 제시되어있다. 이 때, 왼쪽에 있는 방송 클라이언트가 방송하는 장면과 오른쪽에 있는 시청 클라이언트의 방송 시청 장면이 다른 것을 볼 수 있다. 어느 정도 지연시간에 의해 약간의 차이가 나지만 크게 깜박이는 현상 없이 재생이 잘 되는 것을 확인하였다. 5000ms정도의 지연시간은 DMB를 시청할 때와 인터넷 환경에서의 실시간 방송에서도 나타날 수 있는 시간이므로, 이는 실시간 방송이라 봐도 무방하다.



(그림 7) 방송화면(지연시간 5000ms,
왼쪽은 방송 클라이언트 오른쪽은 시청 클라이언트)

5. 결론

본 논문에서는 현재의 인터넷 환경에서의 실시간 방송의 단점을 극복하고자 모바일 환경인 안드로이드 환경에서 단말 대 단말 영상 전송 기법을 제안하였다. 시스템은 서버-클라이언트 모델로 방송 클라이언트(Broadcast Client)가 서버(Server)에게 데이터를 전달하고 서버가 시청 클라이언트(Viewer Client)들에게 데이터를 전달하는 방식이다.

또한 본 논문에서 제안한 기법으로 실시간 방송을 수행한 결과, 무선 인터넷 상태가 안정적일 때 지연시간을 약 5000ms이상 주었을 경우, 깜박임 없이 실시간으로 영상이 전송 되는 것을 확인하였다. 이는 실시간으로 인터넷 방송을 보는 것과 비슷하거나 더 좋은 결과이다. 이에 무선 인터넷 환경이 더 발전하게 된다면 지연시간은 추후 더욱 감소할 것이라고 예상된다.

참고문헌

[1]http://www.hankyung.com/news/app/newsview.php?aid=2010031996551
[2]http://www.kandroid.org/guide/basics/what-is-android.html
[3]언로킹 안드로이드 개발자 가이드, Frank Ableson 외 2인, 프리렉, 2010
[4]RTP, Colin Perkins, Addison-Wesley, 2003
[5]http://www.networksorcery.com/enp/default1001.htm
[6]Data Communications and Networking, Forouzan, McGrawHill, 2007
[7]안드로이드 2 마스터 북, 사이드 하시미, 사티아 코마티네, 제이펍, 2010
[8]http://developer.android.com/index.html