임베디드 시스템 기반 정보가전에서 멀티미디어 스트리밍 플랫폼 설계 및 구현

박승호‡, 정하림§, 정연돈¶
‡고려대학교 컴퓨터 정보통신 대학원
§고려대학교 컴퓨터 · 전파통신공학과
¶고려대학교 컴퓨터 · 통신공학부
e-mail : {hohpark, harim3826, ydchung}@korea.ac.kr

Design & Implementation of a Multimedia Streaming Platform on Embedded System-based Consumer Electronics Devices.

Seungho Park‡, HaRim Jung§, Yon Dohn Chung¶
‡Graduate School of Computer and Information Technology, Korea University
§Dept. of Computer and Radio Communications Engineering, Korea University
¶Division of Computer and Communication Engineering, Korea University

요 약

최근 VOD(Video On Demand) 사이트의 이용시간이 급격이 늘어나면서 온라인 멀티미디어 서비스의 요구가 늘어나고 있다. 하지만, 하드웨어 자원이 빈약한 가전 기기 (CE: Consumer Electronics)에서 이러한 서비스를 제공하기에는 잦은 버퍼링으로 인한 서비스 품질 저하, 애플리케이션 업데이트의 불편함 및 안정성 문제 등 많은 어려움이 있다. 본 논문에서는 동적 버퍼 사이즈 결정 기법을 통하여 버퍼 언더런을 최소화 함으로써 동영상 시청 환경을 최적화 한다. 또한 Ajax 기술 기반의 Web OS 플랫폼을 적용하여 모듈 별 업데이트가 가능하고 시스템 안정성을 향상시키도록 플랫폼을 설계 및 구현하고 그 성능을 확인한다.

1. 서론

하드웨어와 소프트웨어의 기술 발전과 함께 초고속 인터넷 및 무선망 보급이 급격히 이루어 지면서 본격적으로 휴대용 인터넷 기기들이 출시되고 있다. 이와 더불어, TV 와 같은 가전기기(CE: Consumer Electronics)에서도 점점 인터넷 서비스 기능이 추가되고 있다. 한편, 최근 들어 VOD(Video On Demand) 전문 사이트인 Netflix 가 6 개월 만에 온라인 스트리밍 서비스 이용가입자 1 백만 명을 돌파 하였으며, 프리미엄 콘텐트 서비스를 제공하는 Hulu 의 가입자가 1 년 만에 서비스 초기의 9 배 이상 증가하는 등 온라인 스트리밍 서비스의 선호도가 급 상승 하고 있다.

이러한 시장의 요구에 부흥하기 위해 다양한 기기에서 온라인 멀티미디어 서비스 기능을 구현하기 위해 노력하고 있다. 하지만 이러한 기능을 저 성능의 프로세서와 저 용량의 메모리를 탑재한 CE 기기에서 구현하는 데에는 많은 제약과 어려움이 따른다.

본 연구에서는 500 MIPs 의 속도를 가지는 ARM 코어 CPU 와 512MB 의 RAM, 별도의 storage 를 제공하지 않는 사양을 가지고 있는 CE 기기에서 멀티미디어 스트리밍 플랫폼을 구현하고 이 때 발생할 수 있는 제약사항과 극복방안에 대해 논하였다. 플랫폼은 멀티미디어 플레이어와 플레이어를 구성하는 애플리케이션 프레임워크로 이루어져 있다. 별도 스토리지 장치 없이 스트리밍만으로 플레이어를 구성하는 경우, 버퍼 메모리의 핸들링이 성능을 크게 좌우한다. 동영상의 초기 시작이나, 점프 플레이 시 버퍼링 시간을 줄여 사용성을 높여야 하고, 낮은 망 속도에서는 버퍼 핸들링을 통하여 버퍼가 부족한 상황으로의 진입 횟수를 최소화 하여야 한다. 이를 위해 플레이어 스트리밍에 대한 버퍼링 최적화 방법을 적용하여 효과를 거두었다. CE 기기는 펌웨어의 안정성이 중요한 관리 포인트이다. 또한, 인터넷 애플리케이션 특성상 빈번한 업데이트가 필요하다. 소프트웨어의 업데이트를 기존 Native Firmware 방식으로 사용해서는 이 두 가지 요건을 만족시키기가 어렵다. 따라서 본 논문에서는 애플리케이션의 업데이트 편의 및 안정성, 속도 향상을 위한 Ajax 기반의 Web OS [1]구조를 제안한다. Web 표준을 따르고 있으므로 Device 독립적인 개발이 가능하고, 프로그램의 호환성을 보장받을 수 있다.

2. 연구 배경

CE 기기에서 온라인 멀티미디어 서비스를 제공하기 위한 시도는 여러 곳에서 이루어 지고 있다. 야후와 인텔은 TV 위젯 서비스를 위해 제휴를 맺고 인텔 미디어 프로세스 칩이 사용되는 셋톱박스 등을 통해

서비스 제공을 시작하였다. 애플 TV 도 iTunes 를 통하여 VOD 서비스를 제공하고 있다. 이러한 기기들의 공통적인 특징은 고사양의 CPU 와 GPU, 대용량 메모리와 저장장치를 장착한 하드웨어 환경에서 서비스를 제공 하고 있다는 점이다. 이에 반하여 본 연구에서는 별도의 저장장치를 사용하지 않고 한정된 메모리를 지니는 CE 기기들 중 TV 에서 온라인 멀티미디어 서비스를 제공하는 것을 목적으로 한다. <표 1>에 주요 멀티미디어 기기 별 하드웨어 사양을 명시하였다.

<표 1> 멀티미디어 Device HW 사양 비교

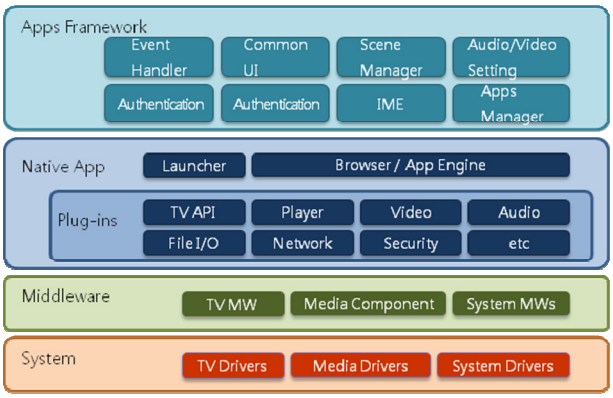
기기	**Gigabyte STB**	**PS3**	**연구에 적용된 TV**
최대 해상도	1920 * 1080	1920 * 1080	1920 * 1080
CPU	Intel CE3100 1.2GHz	3.2 GHz	ARM 500MHz
Memory	768MB DDR2	512MB	256MB
Storage	160GB HDD	120GB HDD	N/A
그래픽 카드	PowerVR SGX535	NVidia 550 MHz	System on

3. 멀티미디어 스트리밍 플랫폼 제안

이 장에서는 서론에서 언급한 제약적인 CE 기기에서 멀티미디어 스트리밍 플랫폼을 제안한다. 이 플랫폼은 멀티미디어를 재생하는 플레이어와, 플레이어를 실행하기 위한 애플리케이션 프레임워크로 구성되어 있다.

3.1. 애플리케이션 프레임워크(Application Framework)

멀티미디어 플레이어를 포함하는 애플리케이션은 자바스크립트 엔진 위에서 Ajax(Asynchronous JavaScript and XML) [2] 로 동작하는 Web App 방식으로 구현 하였다. UI 는 HTML, CSS 로, 데이터 처리는 JavaScript 를 사용하고, Device 에 특화된 확장기능 즉, 멀티미디어 재생, System I/O 등은 Plug-in [3]을 통해 제공한다. 구조는 (그림 1)과 같다.



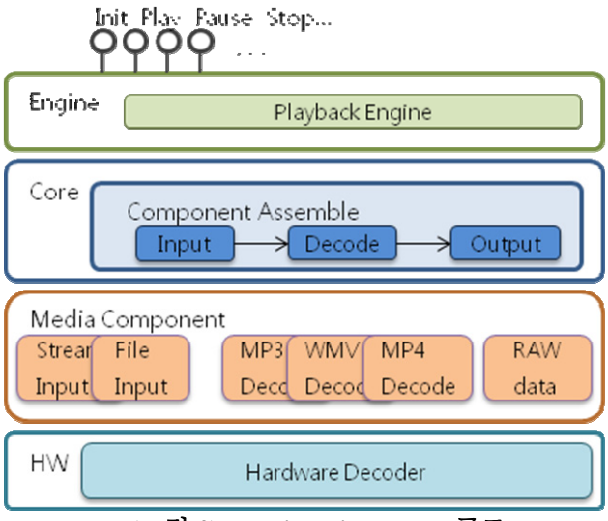
(그림 1) Web App Framework 구조

기존의 CE 기기에서 주로 사용하는 펌웨어 업그레이드 기반의 Native Application Framework 를 그대로 적용하게 된다면, 수시로 일어나는 인터넷 애플리케이션(App)의 업그레이드 요구에 유연하게 대응할 수 없다. 수많은 App 의 변경이 일어날 때 마다 바이너리 코드를 기기에 인스톨 시킨다면, 업그레이드의 불편함, 유지보수의 어려움, 에러 발생시의 대처 등과 같은 문제를 야기할 수 있다. 따라서 스크립트 코드 기반의 Framework 을 적용하여 프로그램 컴파일 없이 단순 스크립트 코드만 다운로드 하면, 업데이트가 가능한 구조를 채택함으로써 앞서 설명한 문제점들을 해결하는 좋은 대안이 될 수 있다. 또한, Web 표준 방식을 따르고 있으므로 브라우저가 탑재된 기기 어디에서나 구동이 가능하기 때문에 다양한 기기에서의 호환성을 보장받을 수 있다. 이는 플랫폼마다 다르게 지원하는 C++, Java, C# 등의 언어에 맞추어 재 개발이 필요 없다는 것을 의미하며, 빠른 개발이 가능함으로써 소프트웨어의 경쟁력 향상을 가져올 수 있다.

시스템 파워가 떨어지고 제약이 많은 CE 기기에서는 성능 향상을 염두 하여야 한다. Ajax 기법은 사용자가 서버 측에 요청을 보내면, 서버의 데이터만 비동기 적으로 변화 시킬 수 있으며, 이렇게 서버와 소량의 데이터만 주고받기 때문에 빠른 응답을 제공한다. 따라서 임베디드 시스템 같은 제한적인 시스템 자원에서 효과적인 속도 문제 해결 방안이 될 수 있다. 멀티미디어에 대한 메타 데이터 및 화면에 대한 정보 등은 XML, 혹은 JSON 형식으로 서버를 통해 내려 받고, UI 등의 동작은 HTML, CSS, JavaScript 로 로컬 디바이스에서 구현하면 사용자 중심의 리치한 UI 설계가 가능해지고, Ajax 의 장점인 유동성 있는 구현이 가능해 진다. 단, 임베디드 기기에 적용이 가능한 브라우저는 완벽한 풀 브라우저가 아닌 제한적인 브라우저를 사용하는 것이기 때문에, 브라우저 엔진이 지원하는 HTML, JavaScript, CSS 태그에 제한이 있다. 또한 Ajax 방식을 사용하면서 XMLHttpRequest 객체 사용에 있어 Cross-Domain 이슈가 남아 있다 [4][5]. 즉, 메타 데이터를 전달해주는 Contents Provider 는 외부에 있고, 데이터를 받아오는 스크립트 코드는 로컬 디바이스에 있기 때문에 디바이스에 설치되어 있는 스크립트 엔진에서 해당 기능이 사용 가능하도록 구현 되어야만 하는 제약 사항이 있다.

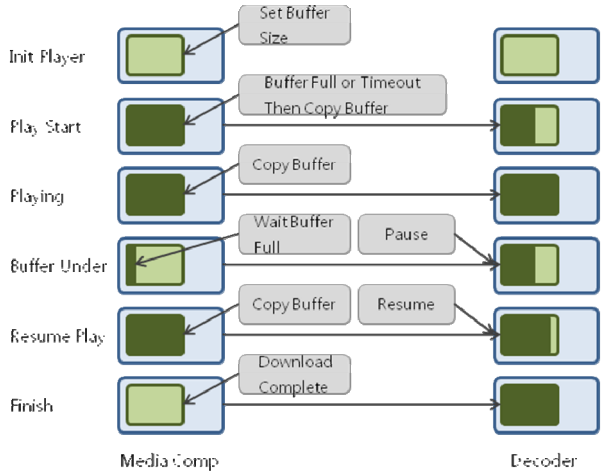
3.2. 멀티미디어 플레이어 (Multimedia Player)

멀티미디어 플레이어는 멀티미디어 엔진, 코어, 미디어 컴포넌트로 이루어져 있다. 멀티미디어 엔진은 재생을 위한 Playback 엔진으로 구성되며 Playback 엔진에 대한 인터페이스를 Application 에 제공한다. 미디어 컴포넌트는 Stream, File 등을 위한 Input 컴포넌트와 MP3, MP4 등의 디코딩을 위한 컴포넌트로 이루어져 있으며, 코어에서는 특정한 멀티미디어 서비스를 제공하기 위한 컴포넌트들의 조합 연결로 이루어진다. 플레이어 구조는 (그림 2)와 같다.



(그림 2) Multimedia Player 구조

저 용량의 한정된 메모리를 가지고 있는 하드웨어 상에서 멀티미디어 스트리밍 재생을 지원하고 재생 품질을 보장하기 위하여 버퍼 핸들링 기법을 사용한다 [6]. 초기 버퍼링 시간은 스트리밍 헤더 정보와 동영상 재생이 시작될 수 있는 최소한의 크기만 받을 수 있도록 하여야 빠른 재생을 통해 사용성이 개선된다. 연결된 미디어 컴포넌트는 각자 데이터를 미디어 데이터 버퍼에 쓰고 읽으면서 데이터를 전달한다. 데이터의 크기는 미디어 컴포넌트에 따라 다르게 되므로 버퍼를 설정하고 관리하는 버퍼 핸들러(Buffer Handler)를 사용한다. 버퍼 운영이 어려운 점은 연결된 버퍼들이 사용하고자 하는 사이즈와 단위가 달라 추가적인 관리가 필요하기 때문이다. 이러한 동작은 버퍼의 사이즈와 단위를 추상화하여 버퍼 핸들러로 모아 사용자가 설정할 수 있도록 편의를 제공한다. 버퍼 핸들링 구조는 (그림 3)과 같다.



(그림 3) 미디어 데이터 버퍼 핸들링 구조

미디어 데이터 버퍼는 스트리밍 데이터를 버퍼링하며 일정한 unit Buffer 집합으로 이루어져 있다. 스트리밍 속도가 느려질 경우 버퍼 핸들러에서 설정한 버퍼 사이즈 보다 데이터가 줄어들게 되면(Buffer Under), (그림 2) 에서 설명한 Playback 엔진은 재생을 일시 정지하고 (그림 3 의 Buffer Under 상태) 일정 양 만큼 버퍼가 채워질 때까지 기다린 후, 설정한 만큼 데이터가 채워지면 다시 재생한다. (그림 3 의 Resume 상태)

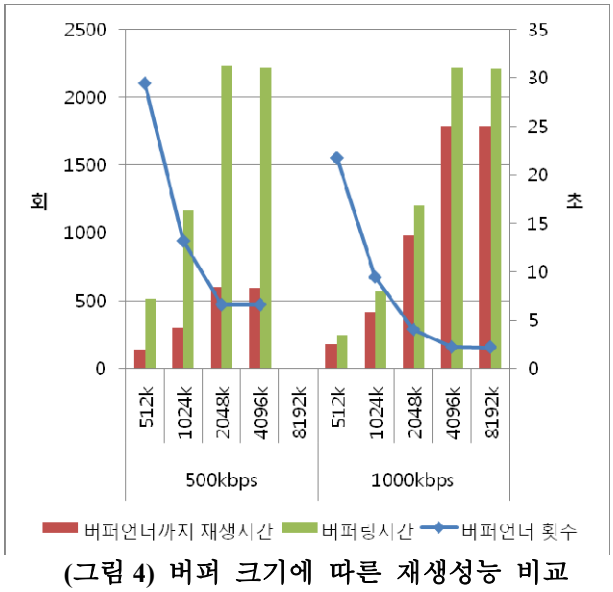
미디어 데이터 버퍼의 초기 사이즈는 재생이 가능한 최소 사이즈, 즉, 멀티미디어의 메타정보와 실제 미디어 데이터가 들어오기 시작하면 재생을 시작하여 버퍼링 대기 시간을 최소화 하여야 한다. 초기버퍼량 결정 기법에 더하여, 가변 하는 인터넷 망 속도에 대응하기 위하여 스트림 다운로드 속도를 모니터링 하면서 미디어의 Bitrate 와 비교하여 그 이하로 속도가 떨어지면 버퍼의 크기를 늘려서 재생할 수 있는 시간을 연장 하도록 적용하였다. 이렇게 하면 버퍼링의 시간은 늘어나지만 늘어난 버퍼만큼 재생 가능한 시간이 증가하므로, Playback 엔진이 Pause 상태로 진입하는 횟수가 감소하게 되어 사용자의 불편을 덜 수 있다. 단, 사용성을 감안하여 버퍼링 시간에 Timeout 을 적용하였다.(본 연구에서는 30 초) 처음부터 버퍼 사이즈를 큰 고정 값으로 잡을 수 있겠으나, 이렇게 할 경우에는 빠른 망 속도에서도 처음 재생에 진입하거나, Jump Play 이후 재생에 진입하는데 버퍼 크기만큼의 데이터 다운로드 시간을 기다려야 하므로 사용성이 떨어진다. 따라서 동적으로 버퍼크기를 조절하는 것이 중요하다. <표 2> 에서는 Bitrate 보다 빠른 망 속도에서(1Mbps) 초기 버퍼 사이즈에 따른 재생 진입 시간의 차이를 보여준다.

<표 2> 버퍼 사이즈에 따른 재생 진입 시간

512KB	1024KB	2048KB	4096KB
3.6 초	8.2 초	17.1 초	35.1 초

4. 구현 및 성능평가

2138kbps 의 Bitrate, 3979 초의 재생시간을 가지는 영상을 500kbps, 1000kbps 의 극단적으로 열악한 망 속도에서 버퍼 크기를 512KB 부터 8192KB 까지 변화시켜가며 재생하여 성능 차이를 측정하였다.



(그림 4) 버퍼 크기에 따른 재생성능 비교

(그림 4) 에서와 같이 미디어의 Bitrate 보다 느린 망 속도에서는 최대한 많은 크기의 버퍼 사이즈를 잡아주는 것이 버퍼 언더 상태로 진입하는 횟수를 줄이고, Pause 상태로 진입하는 데까지 걸리는 재생시간을 늘리는 데 영향을 줌을 알 수 있다.

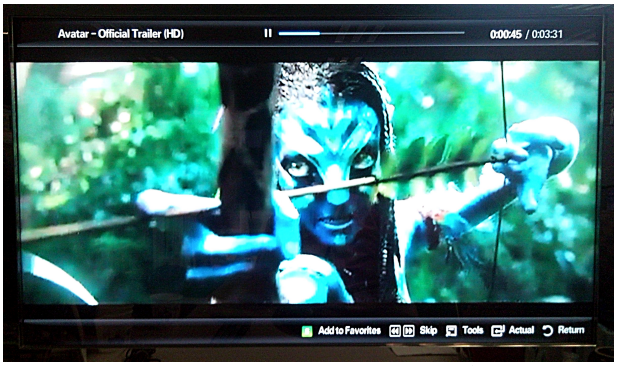
동작 시험 동영상의 속성은 <표 3>과 같다. 온라인 VOD 서비스에서 주로 사용하는 평균적인 샘플 동영상을 사용하였으며, 망 속도는 테스트 영상의 Bitrate 보다 느린 500kbps 의 환경을 조성하였다. (그림 5)와 같이 대화면 Device 에서 실시간 멀티미디어 스트리밍 테스트를 수행하였다. 테스트 결과 <표 4>에서와 같이 열악한 환경에서도 동작 성능이 개선됨을 볼 수 있었다.

<표 3> 테스트 동영상 속성

	Video1	Video2
File Size (KB)	97,452	13,137
Bitrate (kbps)	622	594
Resolution(Pixel)	480 * 270	470 * 270
Total Time (Sec)	1,281	181

<표 4> 버퍼 핸들링을 통한 성능 개선

		개선 전	개선 후
Video1	버퍼 사이즈	512kB 고정	2048kB 변경
	버퍼 언더 진입 횟수	213 회	53 회
	버퍼 언더 까지 평균 재생시간	6 초	23.8 초
Video2	버퍼 사이즈	512kB 고정	2048kB 변경
	버퍼 언더 진입 횟수	7 회	1 회
	버퍼 언더 까지 평균 재생시간	23.8 초	99 초



(그림 5) Multimedia App 실행 화면

5. 결론 및 향후 연구 과제

본 논문에서는 제한적인 자원의 CE 기기에서의 제약 사항을 알아보고 그 극복 방안으로 Ajax 기반의 Web App Framework 와 버퍼 핸들링 기술을 적용한 Multimedia Player 를 이용하여 멀티미디어 스트리밍 플랫폼을 설계 및 구현 하였다. 이로 인한 장점은 시스템의 안정성을 향상 시키고, 다양한 기기에 빠른 적용이 가능하며, 낮은 망 속도에서도 스트리밍 환경의 최적화가 가능함을 확인 하였다.

본 논문에서 제안한 데이터의 크기, 인터넷 속도에 따라 설정하는 버퍼링 핸들 기술 이외에 최근에는 Live Streaming 기술을 이용하여 망 속도에 대응하여 동적으로 화질을 바꾸는 기술이 차츰 선보이고 있다. [7] 향후에는 멀티미디어 플랫폼에 이 기술을 적용하면 더욱 유연한 재생 환경을 제공할 수 있다.

본 연구에서는 멀티미디어 재생 및 Device System 서비스를 접근하기 위하여 Plug-in 방식을 사용하여 구현하였고, 이는 App 작성시 브라우저에서 동작하는 Plug-in 을 Device 에 설치해야 함을 의미한다. 하지만, 향후 HTML5 [8] 전체 스펙을 지원하는 브라우저가 늘어나면 이러한 멀티미디어 Play, 데이터 저장, I/O 등의 System 동작까지 표준화된 API 를 통한 지원이 가능한 구조가 된다. 앞으로 이러한 HTML5 를 CE 기기에 적용하기 위한 추가 연구가 필요하다.

참고문헌

[1] G. Lawton, “Moving the OS to the Web,” Computer, vol. 41, no. 3, pp. 16–19, 2008.

[2] Deitel Harvey M, Deitel Paul J, “AJAX, Rich Internet Applications and Web Development for Programmers”, Prentice Hall Ptr, 2008.

[3] Mozilla “Plug-in Basics” https://developer.mozilla.org/en/Gecko_Plugin_API_Reference/Plug-in_Basics

[4] Mozilla “Same origin policy for JavaScript”, https://developer.mozilla.org/en/Same_origin_policy_for_JavaScript

[5] Collin Jackson, Helen J. Wang, “Secure Cross-Domain Communication for Web Mashups”, Proceedings of the 16th international conference on World Wide Web, pp.611-620, 2007

[6] 서광덕, 정순흥, “프로그레시브 멀티미디어 스트리밍 서비스를 위한 초기 버퍼링 시간 결정 기법”, 정보과학회논문지:컴퓨팅의 실제 및 레터, 제 14 권 제 2 호, pp. 206-210, 2008.4

[7] Apple “Http Live Streaming OverView”, http://developer.apple.com/iphone/library/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/Introduction/Introduction.html

[8] I. Hickson, “HTML 5”, Technical report, Web Hypertext Application Technology Working Group HTML 5, 2010. Working Draft, http://www.whatwg.org/specs/web-apps/current-work.