동적 UI 서비스를 위한 RUI(Remote UI)의 확장

추길호*, 권오영*, 김태근**
*한국기술교육대학교 전기전자공학과
**유비코드
e-mail:flyhigh80@kut.ac.kr

Extension of RUI (Remote UI) for the Dynamic UI Service

Gil-Ho Chu*, Oh-Young Kwon*, Tae-Geun Kim**
*Dept of Electric & Electronic Engineering, Korea University of Technology and Education
**UBICOD Co., Ltd.

요 약

UPnP 기반의 Intel® RUI는 서버의 자원을 이용하여 클라이언트에 UI 서비스하고 사용자의 입력을 받아 처리한다. 본 논문에서는 동적 UI 서비스시에 발생하는 '변화가 빈번한 이미지 전송 문제' 및 '서비스 대상 프로그램에서 출력되는 사운드 전송 문제'에 대한 해결 방안을 제시한다. 또한 Flash Content를 서비스하는 RUI Server/Client 구현을 통해 검증하여 실제 적용 가능성을 보여준다.

1. 서론

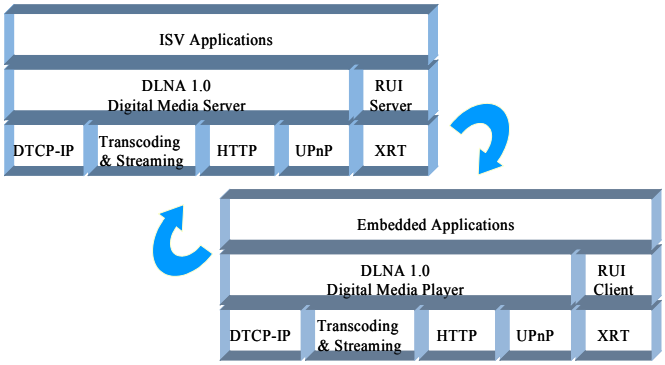
정보기술의 발달과 메인 컨트롤러의 가격 하락으로 가전기기는 단순한 기계를 벗어나 컴퓨팅 파워를 갖는 기계가 되었으며 네트워킹이 가능해지면서 타 기기와의 상호운용성을 갖게 되었다. 특히 가정 PC의 고사양화로 인해 PC를 홈서버로 이용할 수 있게 되면서 홈서버와 가전기기의 결합을 통한 홈 엔터테인먼트에 대한 관심이 높아졌다. 홈 엔터테인먼트 가전기기의 한 예로 Settop Box을 들 수 있다. 하지만 저가의 Settop Box의 성능으로는 사용자의 요구사항을 만족시킬 수 없는 상황이다. 이와 같은 문제를 해결하기 위해 홈서버의 컴퓨팅 자원을 이용할 수 있는 방안을 생각해 볼 수 있다. VNC, 마이크로소프트의 원격 데스트톱 및 RUI는 서버의 자원을 이용하여 클라이언트의 사용자입력 처리 및 서버에서 실행되는 프로그램의 UI를 전송하는 기능을 제공한다[1,2,3]. 이 중 RUI는 UPnP를 기반으로 하기 때문에 다른 디바이스와의 연결이라는 점에서 유연성을 제공하며 개발이라는 관점에서 볼 때 손쉽게 제어할 수 있는 이점을 가지고 있다. 또한 Intel에서는 Intel® NMRP를 정하고 표준안을 만들어 제공하고 있다[1].

RUI를 사용함으로써 클라이언트는 적은 컴퓨팅 자원으로 홈서버에서 제공하는 UI 서비스를 사용할 수 있다. 이런 기능을 이용하면, 서버에서 실행되는 웹 브라우저의 화면을 클라이언트로 전송하여 클라이언트에서는 웹 브라우저 없이 인터넷을 사용할 수 있다. 하지만 UI의 배경화면 또는 인터페이스가 빈번하게 변하는 동적 UI의 경우 컴퓨팅 파워가 약한 임베디드 시스템에서는 전송된 화면에 대해 실시간 처리를 하지 못하는 경우가 발생한다. 이 논문에서는 동적 UI 서비스를 위하여 기존의 RUI의 문제점 분석하여 대안을 제시하고 확장하였다. 2장에서는 Intel® RUI의 동작 및 문제점에 대해서 살펴보며 이 문제점에 대한 대안을 3장에서 제시한다. 제시된 대안을 바탕으로 4장에서 구현 및 검증을 한다.

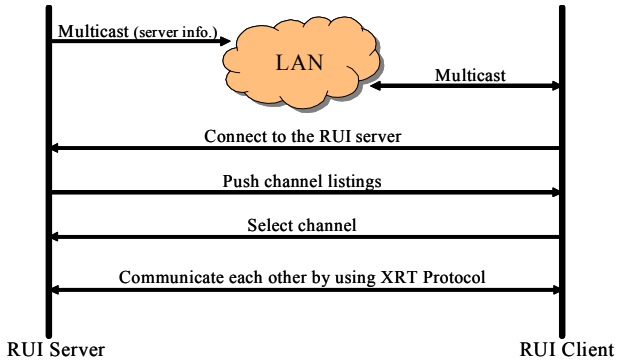
2. Intel® RUI의 동작 및 문제점

RUI는 (그림 1)과 같은 구조로 갖으며 (그림 2)와 같이 동작한다. 멀티캐스트를 이용하여 디바이스

를 찾으며 클라이언트는 Server로 부터 채널리스트를 전송받아 원하는 채널의 서비스를 받게 된다[9]. RUI에서 사용하는 XRT Protocol은 원격 디바이스에 이미지를 그리기 위해 DRAWIMAGE명령어, 사용자 입력을 위한 Key와 Mouse 관련 명령어, Media 재생을 위한 명령어 등을 지원한다[4].



(그림 1) Intel® RUI 구조



(그림 2) Intel® RUI 작동 구조

2.1. 이미지 전송의 문제점

RUI의 XRT 프로토콜은 이미지 전송을 위해 DRAWIMAGE 명령어를 제공한다[10]. 이 명령어는 JPEG과 PNG 포맷을 지원한다. 이미지를 이와 같은 포맷으로 전송할 경우 전송량의 측면에서 많은 이점을 얻을 수 있다. 압축된 이미지를 디코딩 후 프레임버퍼 등을 이용하여 화면에 출력할 경우 소요되는 시간은 <표 1>과 같다.

테스트는 Pentium Ⅲ 600MHz PC에서 이루어졌다. <표 1>의 결과로 볼 때 640x480 이미지 1장을 처리하는데 120㎳가 소요되므로 분당 8프레임의 성능이 나온다.

<표 1> 클라이언트에서의 JPEG 출력 시간

측정항목	320x240일 경우	640x480일 경우
JPEG 디코딩	14000㎲	54000㎲
Blit	8500㎲	32800㎲
Flip	34000㎲	34000㎲

만약 네트워크 전송 시간 및 클라이언트 내부 처리 소요시간을 고려한다면 초당 처리할 수 있는 프레임 수는 더 낮아진다. 실제 클라이언트는 테스트한 PC보다 성능이 낮기 때문에 수행시간이 더 길어질 것으로 예상할 수 있다. 따라서 정적 UI의 서비스는 가능하나 동적 UI의 서비스는 어렵다는 것을 알 수 있다. 또한 이 측정 결과에서 주목해야 할 부분은 Flip 수행시간이다. 이 시간은 처리된 이미지를 화면에 업데이트하는 디바이스의 성능과 밀접하게 관련이 있기 때문에 서버에서 전송할 이미지의 크기 또는 용량을 조절하여도 향상시킬 수 없다. 따라서 동적 UI 서비스를 위해서는 Blit와 Flip 시간을 줄여야 한다.

2.2. 사운드 전달 방식의 부재

동적 UI 서비스에서 고려되어야 할 또 다른 부분은 사운드 전송이다. 사운드는 동적 화면에 효과를 더 해준다. XRT 프로토콜에서는 미디어 관련 명령어를 제공하여 클라이언트에서의 미디어 Playback 기능을 제공한다. 하지만 [4]에서 확인할 수 있는 것처럼 이미 만들어진 음악 파일의 서비스에는 효과적이지만 서비스되는 특정 프로그램에서 발생하는 사운드의 전송에는 사용할 수 없다. 따라서 프로토콜의 확장이 필요하다.

3. Intel® RUI의 확장

3.1. 이미지 전송

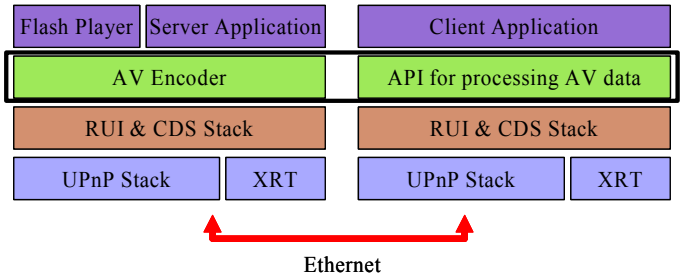
<표 1>을 통해 기존의 방식은 동적 UI 서비스에 적합하지 않음을 알 수 있었다. 이 문제에 대한 대안으로 Settop Box에 탑재되어 있는 A/V 디바이스의 사용을 생각해 볼 수 있다. 하드웨어를 이용하여 디코딩 후 출력할 경우 앞서 문제로 제시한 Blit와 Flip에 걸리는 시간문제를 해결할 수 있다. 물론 실시간으로 이미지를 인코딩해야 하는 서버의 부담을 고려해야 할 것이다. 또한 JPEG 파일보다 프레임당 사이즈가 커짐으로써 발생하는 문제도 고려하여야 한다.

3.2. 사운드 전송

서비스될 프로그램의 사운드 출력을 실시간으로 추출하여 서비스하기 위해서는 PLAYAUDIO와 같은 프로토콜을 추가하는 방안을 생각해 볼 수 있다. 또한 추출된 사운드 데이터는 사이즈가 큰 RAW데이터이기 때문에 이 데이터를 클라이언트에서 하드웨어 디코딩이 가능한 포맷으로 변환하여 전송하는 방식을 고려해야 한다.

3.3. 수정된 Intel RUI의 구조

앞의 대안을 바탕으로 RUI의 구조를 (그림 3)과 같이 변경할 수 있다. 각 단에 AV 처리를 위한 부분을 추가하였다.



(그림 3) 수정된 RUI 구조

이 논문에서는 A/V 인코딩을 위해 MPEG2 인코더와 MPEG1 Audio Layer2 인코더를 사용하며 서비스될 프로그램으로는 Flash Player를 이용한다.

4. 구현

구현은 Intel®의 UPnP Remote UI 소스를 기반으로 하였으며 서버는 WindowsXP에서 Visual C++ 2003을 이용하였다. 클라이언트는 리눅스 환경에서 gcc, SDL 라이브러리 및 AV 디바이스 API를 이용하였다. MPEG2 인코더는 MPEG Software Simulation Group의 소스를 사용하였으며 MP2 인코딩을 위해 twoLame 라이브러리를 이용하였다[5,6].

4.1. MPEG2 지원

본 논문에서 사용하는 클라이언트 Settop Box에서 지원되는 포맷은 MPEG2 이다. MPEG2 인코더는 소스가 공개되어 있으며 사용이 간편한 이점이 있다[5]. 적용에 앞서 기존 방식인 JPEG과 비교해 볼 필요가 있다. <표 2>는 JPEG과 MPEG2의 인코딩 시간을 비교한 표이다.

<표 2> 인코딩 시간 비교 (단위:sec)

	JPEG	MPEG2
PIV 2.4GHz	0.038	0.085
AMD 3000+	0.027	0.091

<표 2>에서 보는 것과 같이 인코딩 시간은 JPEG이 우수하다. MPEG2의 경우 모션과 관련된 작업이 이루어지기 때문에 상대적으로 시간이 많이 걸림을 예측할 수 있다. AMD 3000+의 경우를 볼 때 약 10fps 의 속도를 보임을 알 수 있다. JPEG의 경우 서버에서의 인코딩 속도는 빠르나 2.1절에서 본 것과 같이 클라이언트에서 8fps 정도의 처리 성능을 갖기 때문에 서버에서의 빠른 인코딩은 의미가 없다. 하지만 MPEG2의 경우 서버에서의 인코딩 시간은 길지만 클라이언트에서 하드웨어를 이용한 디코딩을 할 경우 서버의 최대 성능만큼 초당 처리 프레임 수를 높일 수 있다. 여기에서 발생할 수 있는 문제점은 서버의 최소사양의 기준이다. 10fps 정도로 인코딩을 해주어야 동적인 UI의 서비스가 가능하기 때문에 실험에 사용한 두 PC 보다 성능이 낮은 PC는 적합하지 않을 수 있다. 최근 고사양을 요구하는 많은 게임의 보급으로 가정의 PC 사양이 많이 높아진 상황이긴 하나 그렇지 못한 가정의 PC에서는 이와 같은 서비스가 힘들 것으로 예상된다.

또한 인코딩된 데이터의 크기를 비교해 볼 필요가 있다. 720x480 크기의 이미지를 이용하였으며 결과는 <표 3>과 같다.

<표 3> 프레임 사이즈 비교

	JPEG	MPEG2
크기	30656byte	41715byte

MPEG2의 경우 모션 정보를 포함하기 때문에 JPEG에 비해 데이터의 크기가 큼을 알 수 있다. 프레임 단위의 처리를 기본으로 하기 때문에 B, P 프레임은 생성하지 않는다. 이미지의 상관도에 따라 프레임당 크기는 변하겠지만 JPEG과 큰 차이는 없다.

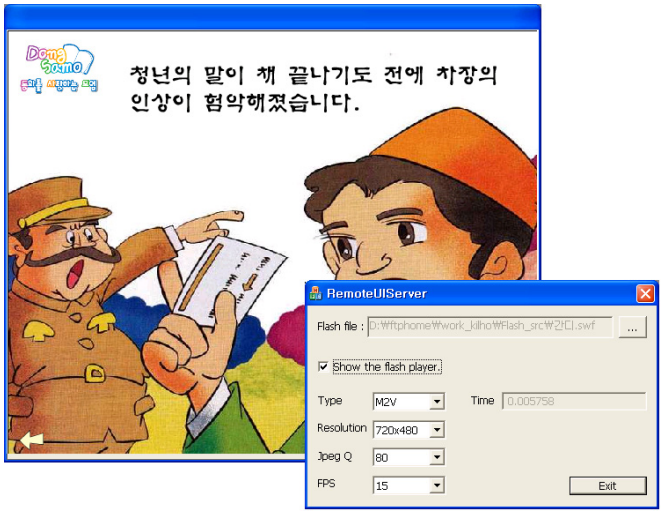
Flash Player에서 서버 성능에 맞게 주기적으로 화면 이미지를 얻어 MPEG2 인코딩을 한다. 이렇게 생성된 이미지를 DRAWIMAGE를 이용하여 전송한다. 클라이언트는 전송된 이미지의 포맷을 확인 후 각 포맷에 맞게 처리를 한다.

4.2. 사운드 전송

서비스될 프로그램의 사운드를 추출하기 위해서는 각 프로그램 별로 작업이 이루어져야 한다. 본 논문의 구현에서 사용할 Flash Player의 경우 사운드를 추출하기 위하여 API 후킹기술을 이용한다. 후킹을 통해 추출된 RAW 데이터를 MP2로 압축하여 새로 정의한 PLAYAUDIO 프로토콜을 이용하여 클라이언트로 전송한다. 클라이언트는 별도의 디코딩 작업없이 A/V 디바이스를 이용하여 사운드를 재생한다. 사운드의 압축 옵션은 사운드의 성질에 따라 다르게 하여야 할 것이다. 본 구현의 경우 Flash Player에서 추출된 사운드 데이터는 44.1KHz 16bit Stereo였으며 mp2 128bitrate으로 압축하여 전송하였다.

4.3. 동작 실험

(그림 4)는 구현된 RUI Server이다. 서비스할 Flash Content를 선택 및 전송 포맷에 따른 성능 테스트를 위해 옵션을 제공하며 인코딩 시간을 확인할 수 있도록 하였다. 서버에서 재생되는 flash content의 이미지와 사운드를 추출하여 클라이언트로 전송을 한다.



(그림 4) 제작된 Remote UI Server

실제 266MHz(300MIPS) 급의 Settop Box에 클라이언트를 설치하고 서비스를 했을 경우 0.5sec~1sec 정도의 delay를 보이며 원활하게 작동되는 것을 확인 하였다. 여기서 발생하는 delay는 인코딩, 네트워크 전송 및 디코딩 단계에서 발생하는 것으로 최소화해야 할 부분이며 어느 정도의 delay는 감수해야 한다. 본 논문의 구현에서는 MPEG2 인코더의 최적화는 배제하였다. 최적화의 정도에 따라 delay 문제에 대한 해결이 가능하며 또한 4.1절에서 언급한 것과 같이 I 프레임으로만 처리를 하였기 때문에 인코딩 성능에 따라 P, B 프레임의 사용으로 전송 데이터양을 줄일 수 있다.

(그림 5) Remote UI Client의 화면

5. 결론

본 논문에서는 홈 엔터테인먼트 솔루션 중의 하나인 Settop Box에서 동적 UI 서비스를 위한 Intel® R UI의 확장에 대한 제안을 했다. 동적 UI 서비스시에 발생되는 문제점을 분석하고 해결 방안을 제시 및 구현하였다. 본 논문은 MPEG2 인코더 성능에 대한 고려가 이루어지지 않았기 때문에 서버의 성능이 낮을 경우 서비스가 어렵다고 할 수 있다. 4.3 절에서 언급한 인코더의 최적화에 대한 연구가 필요하며 이 논문에서는 다루지 않은 이미지 및 사운드와 사용자 입력의 동기화 문제에 대한 연구가 계속되어야 한다.

참고문헌

[1] Intel Corp, Intel® Networked Media Product Requirements version 2.1, 2005.02

[2] RealVNC, http://www.realvnc.com/

[3] Microsoft Remote Desktop Protocol http://www.microsoft.com/technet/prodtechnol/Win2KTS/evaluate/featfunc/rdpfperf.mspx

[4] Intel Corp, Extended Device Remote Transfer Protocol(XRT Protocol) version 2.2, 2005.02

[5] MPEG Software Simulation Group http://www.mpeg.org/MSSG/

[6] twoLame, www.twolame.org

[7] Intel® Developer Network for the Digital Home http://www.intel.com/technology/dhdevent/

[8] Remote UI Client and Server V 1.0 http://www.upnp.org/standardizeddcps/remoteui.asp

[9] UPnP Forum, http://www.upnp.org