내장형 리눅스 기반 이동 단말기에서의 MPEG-4 오디오 스트리밍 재생기의 구현†

(Development of MPEG-4 Audio Streaming Player on Mobile Terminal with Embedded Linux Processor)

차 경 애*
(Kyung-Ae Cha)

요 약 본 논문은 내장형 리눅스 기반의 이동 단말기 상에서 MPEG-4 AAC 데이터를 실시간으로 재생하는 소프트웨어를 구현하고 그 실험 결과를 통해서 오차 범위 내에서 MPEG-4 오디오의 재생이 가능함을 검증한다. MPEG-4 AAC 데이터는 압축효율이 높고 음질이 뛰어나 다양한 기기에서 사용이 용이하다. 특히 휴대폰, PDA 등과 같은 이동 환경에서 사용되는 단말기가 급증하면서, 오디오 데이터의 응용어플리케이션 개발의 필요성도 증대되고 있다. 그러나 소형의 이동 단말기는 전력, 메모리 등의 자원의 한계로 인하여 디코딩 과정이 복잡한 MPEG-4 AAC 데이터를 를 재생하는 응용어플리케이션의 개발을 위해서는 단말기의 성능에 최적화된 형태로 소프트웨어 모듈을 구현해야 한다. 이를 위해서 MPEG-4 AAC 디코딩 연산 과정을 단말기의 프로세서 성능에 알맞은 형태로 연산변형하고 스트리밍 서버를 통해서 전송되는 오디오 데이터를 재생할 수 있도록 설계하였다.

핵심주제어 : MPEG-4 Audio, 스트리밍 서비스, 내장형 시스템

Abstract In this paper, we develop MPEG-4 AAC streaming player on embedded Linux processor such as mobile terminals. Moreover we show the experimental results that the player preforms the decoding processes of MPEG-4 AAC data effectively. MPEG-4 AAC technology supports a wide range encoding rates and high sound quality so it is appropriate to adopt various applications. In particular, the need in the development of the application of audio data increases according to significantly increase in devices used in mobile environments, such as cell phones and PDAs. In this environment, it is necessary to optimize the decoding processes to the ability of the terminal hardware in order to play audio data without delays. We also implement the decoding module to optimize the processor capabilities and make the player to decode and play streaming audio data from streaming server.

Key Words : MPEG-4 Audio, Streaming Service, Embedded System

1. 서 론

이동 컴퓨팅을 가능하게 하는 휴대형 단말기는 소형화 및 경량화된 하드웨어로 구성됨에 따라 메모리, 전력 등의 자원이 매우 한정적이다. 반면에 정보의 멀티미디어화와 휴대 단말기기의 수요가 증가함에 따라서, 멀티미디어 데이터를 활용하는 응용어플리케이션의 수요가 매우 높다[1,2]. 특히 오디오 데이터는 영상 정보에 비해서 소형 단말기에서의 활용이 용이하여, 다양한 오디오 재생기가 개발되고 있다. MPEG-4 AAC 오디오 데이터는

† 이 논문은 2005학년도 대구대학교 학술연구비 지원에 의한 논문임.
* 대구대학교 정보통신공학부

압축효율이 높고 음질이 뛰어나 다양한 기기에서 사용이 용이하며, 다양한 채널에서 서로 다른 샘플율(8~96kHz)과 비트율(8~640kbps)을 제공한다. 이는 인코딩 시 사용자의 선택에 따라 음질과 다양한 대역폭을 가질 수 있어 적용분야의 다양성을 가질 수 있음을 의미한다[3]. 즉 MPEG-4 AAC는 소형 이동단말기에서의 오디오 응용어플리케이션에서 활용이 편리하다. 이러한 활용성을 배경으로 본 논문에서는 내장형 리눅스 기반의 소형 단말기에서 스트리밍 서버로부터 실시간으로 전송되는 MPEG-4 오디오 데이터의 재생이 가능한 스트리밍 재생기를 설계하고 개발한다.

현재 일반 PDA에서 제공되는 MPEG-1 재생기와 MP3 재생기의 경우도 IPP 라이브러리를 통해 정수형으로 변환된 함수를 이용함으로써 재생이 가능하다. IPP 라이브러리는 인텔(Intel)에서 제공하는 멀티미디어 데이터 표현을 위한 툴로서 부동소수점 연산 프로세서가 없는 특정 프로세서에 대한 실수 연산 처리를 위해 소프트웨어로 최적화하는데 필요한 라이브러리이며 모든 자료형을 정수로 표현한다. 하지만, MPEG-4 비디오 코덱과 MPEG-4 AAC는 부분적으로 구현되어 완벽한 지원이 이루어지지 않고 있다[4].

소형의 내장형 시스템의 단말기는 일반 PC와는 달리 내장형 CPU를 사용하며, 이러한 내장형 CPU는 회로의 복잡도, 발열 문제와 소비 전력 문제 등으로 인하여 부동 소수점 연산 프로세서(Floating-point Processing Unit, FPU)가 없다[5,6]. 이러한 기기에서 연속적으로 다수의 실수 연산의 처리를 필요로 하는 멀티미디어 데이터를 디코딩하기에는 처리 속도가 늦다. 따라서 구현상에서 부동 소수점 연산 프로세서에 대한 문제점을 하드웨어 칩 또는 어셈블러를 사용하여 해결함으로써 하드웨어에 의존적으로 구현되어진 경우가 많다. 이는 RISC 구조를 가지는 대부분의 내장형 CPU와는 맞지 않다. 이를 해결하기 위해서 MPEG-4 AAC 디코더에서 실수 연산을 필요로 하는 각각의 알고리즘을 정수 연산으로 변형하고 실시간 재생이 가능하게 한다.

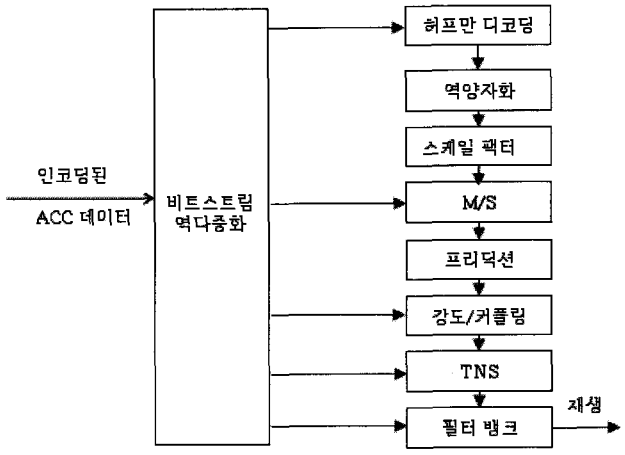
본 논문의 2장에서는 MPEG-4 AAC 디코더의 연산과정을 간략히 설명하고 내장형 시스템에 최적화 연산을 수행하기 위한 연산변형 알고리즘을 제안한다. 3장에서는 연사변형 알고리즘을 적용한 스트리밍 재생기의 구조와 구현방법을 설명하며, 4장에서는 구현된 재생기의 실험결과를 토대로 실시간 스트리밍 재생이 가능함을 보인다. 5장에서 본 논문의 결론과 향후 연구방향을 제시한다.

2. MPEG-4 AAC 디코더의 최적화 연산 설계

2.1 MPEG-4 AAC 디코딩 연산 과정

MPEG-4 AAC는 필터 뱅크(filter bank), 역방향-적응형 예측, 조인트(joint) 채널 부호화, 허프만 부호화 등을 적용하여 뛰어난 신호의 압축 기능을 제공할 뿐만 아니라, 음질을 향상시켰다[3]. 그림 1은 MPEG-4 AAC 복호화기의 블록도이며, 각각의 요소는 MPEG-4 AAC의 툴에 해당한다.

MPEG-4 AAC 디코딩 과정에서의 필터 뱅크 부분의 연산 복잡도에 의해서 자원의 한계가 있는 내장형 단말기에서는 실시간 재생이 어렵다. 특히 필터 뱅크에서는 주파수 영역 신호를 시간 영역의 신호로 변환시키는 IMDCT 연산을 수행하게 되는데, 허프만 디코딩과 필터뱅크의 연산량이 전체의 80% 이상을 차지하며, IMDCT 연산량은 전체 연산량의 약 28.4%를 차지한다[7]. 따라서 자원이 한정적인 단말기에서의 MPEG-4 AAC 데이터의 실시간 재생을 지원하기 위해서는 필터 뱅크의 연산과정을 단말기의 성능에 최적화하는 모듈의 개발이 필요하다. 본 논문에서는 실수연사과정을 정



(그림 1) MPEG-4 AAC 복호화기의 블록도

수로 변환하면서 동시에 최소한의 오차범위를 갖게 하는 알고리즘을 바탕으로 디코딩 모듈을 구현한다.

2.2 IMDCT의 최적화를 위한 연산 변형

MPEG-4 AAC의 인코딩된 데이터는 주파수를 정의역으로 가지며 블록 타입에 따른 데이터의 개수를 N이라고 했을 때, 디코딩 과정은 주파수 영역의 N/2개의 데이터를 실수부과 허수부로 나누어 IMDCT 알고리즘을 통해 N개의 시간영역의 데이터로 바꾼다. 이는 변환 타입에 따라 시작 윈도우(start window), 일반적인 윈도우(normal window), 정지 윈도우(stop window)와 짧은 윈도우(short window)를 가진다.

식 (1)은 MPEG-4 AAC 데이터의 디코딩에 필요한 IMDCT(Inverse Modified Discrete Cosine Transform)연산식이다[7,8]. 주파수 영역이 정의역인 함수 $X(k)$를 시간 영역이 정의역인 함수 $\hat{x}(n)$으로 복원한다.

$$\hat{x}(n)=\frac{2}{N}\sum_{k=0}^{\frac{N}{2}-1}X(k)\cos\left[\frac{\pi}{2N}(2n+1+\frac{N}{2})(2k+1)\right] \quad (1)$$
$$n=0,1,...,N-1$$

이 연산식은 Pre-twiddling, IFFT(Inverse Fast Fourie Transform), Post-twiddling의 세 단계로 분리하여 구현된다.

먼저 Pre-twiddling에서 수행되는 연산식은 식 (2)와 같다.

$$X'(k)=X(k)(\cos(\frac{2k\pi n_0}{N})+jsin(\frac{2k\pi n_0}{N})) \quad (2)$$
$$j=\sqrt{-1}, n_0=\frac{1}{2}+\frac{N}{4}$$

식 (2)에서 N은 윈도우 블록의 크기를 나타내며 디코딩 시에 윈도우의 종류에 의해서 미리 결정되어진다. N의 값이 미리 결정되면, 삼각함수의 값은 k에 의해서 반복되어 나타나는 값으로 미리 결정될 수 있다. 또한 $X(k)$는 입력값으로서 인코딩된 데이터의 실수영역의 값과 허수영역의 값을 각각 $a(k)$와 $b(k)$라고 할 때 $X(k)=a(k)+jb(k)$와 같이 표현된다. 이를 식 (2)에 대입하면 식 (3)과 같다.

$$X'(k)=a(k)\cos(\frac{2k\pi n_0}{N})-b(k)\sin(\frac{2k\pi n_0}{N}) + j(a(k)\sin(\frac{2k\pi n_0}{N})+b(k)\cos(\frac{2k\pi n_0}{N})) \quad (3)$$

식 (4)에서와 같이 pre-twiddling의 연산식은 삼각함수로 인해서 실수연산을 필요로 하게 된다. 이를 정수형 연산과정으로 변형함으로써, 실시간 재생이 가능하게 하도록 다음과 같은 연산으로 변형한다.

이미 살펴본 바와 같이 $\cos(\frac{2k\pi n_0}{N})$와 $\sin(\frac{2k\pi n_0}{N})$에서 윈도우 크기 N이 결정되면 식 (2)에서 보이는 바와 같이 n_0의 값도 결정되기 때문에 모든 k에 대해서 삼각함수의 값도 미리 결정된다. 이를 최소한의 오차로 하는 정수로의 변환을 위해서는 정수범위에 해당하는 값 중 가장 큰 값을 곱해주어야 한다.

우선 식 (3)에서 우변의 실수부와 허수부 값이 존재해야하는 범위를 고려하면 n비트 정수형일 경우 -2^{n-1}과 $2^{n-1}-1$ 사이이다. 이 조건을 항상 만족하도록 실수부와 허수부의 각 항의 범위가 $2^{n-2}-0.5$보다 작거나 같다고 가정한다.

이 때 두 항인, $a(k)\sin(\frac{2k\pi n_0}{N})$와 $b(k)\cos(\frac{2k\pi n_0}{N}))$ 항이 각각 $2^{n-2}-0.5$보다 작다면 정수범위의 최대값을 초과하는 경우가 발생하지 않는다. 따라서 실수부와 허수부의 각 항의 삼각함수값을 정수로 변환하는 과정에서 오차를 최소로 하기 위해서 곱해야 할 값은 $2^{n-2}-0.5$가 된다. 여기서, 0.5의 값은 무시하도록 한다.

이와 같은 가정을 바탕으로 식 (3)을 아래 식 (4)와 같이 변형한다.

$$a(k)\times\cos(\frac{2k\pi n_0}{N})=(2^{n-2}\times\cos(\frac{2k\pi n_0}{N}))/(2^{n-2}/a(k)) \quad (4)$$

$\cos(\frac{2k\pi n_0}{N})$의 값은 윈도우 크기 N에 의해 미리 알 수 있기 때문에 식 (4)에서 $2^{n-2}\times\cos(\frac{2k\pi n_0}{N})$의 값은 모든 k에 대해서 미리 계산할 수 있는 정수값이다. 또한 $a(k)$와 $b(k)$값은 인코딩 된 데이터로부터의 입력값이며, 정수형으로 표현 가능하다. 그러므로 식 $2^{n-2}/a(k)$는 정수 연산으로 수행된다. 이와 같이 식 (4)는 미리 계산된 분자의 정수값을 분모의 정수 연산 결과 값으로 나누는 연산식이 된다.

위와 같은 방법으로 식 (4)에서 보이는 $b(k)\sin(\frac{2k\pi n_0}{N})$, $a(k)\sin(\frac{2k\pi n_0}{N})$, $b(k)\cos(\frac{2k\pi n_0}{N})$ 값에 대해서도 정수 연산을 수행할 수 있다.

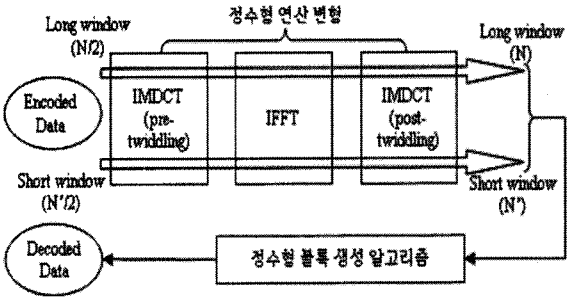
IFFT 과정과 Post-twiddling과정을 위한 연산식도 이와 마찬가지로 삼각함수의 값을 미리 정수형으로 변환하고 입력값인 복소수의 실수부와 허수부를 각각 정수로 표현하여 연산 할 수 있다. [8]을 통해서 자세한 연산과정을 알 수 있다.

3. 내장형 리눅스 기반 MPEG-4 AAC 스트리밍 재생기의 구현

3.1 이동단말기에 최적화된 연산변형 디코딩 모듈 구현

그림 2는 MPEG-4 AAC 디코더의 실수연산을 정수형 연산으로 변형한 알고리즘을 이용한 재생기의 디코딩 과정을 보인다.

입력값인 인코딩된 데이터는 윈도우 타입에 따라서 N개의 데이터를 가지며, 디코딩과정은 주파수 영역의 데이터를 실수부와 허수부로 나누어 2장에서 제안한 정수 연산과정으로 변형된 IMDCT 연산과 IFFT 연산을 통해서 N개의 시간 영역의 데이터로 디코딩한 후 정수형 블록 생성 알고리즘을 통해서 각각 생성된 윈도우의 중첩 과정을 거쳐서 오디오를 재생한다.

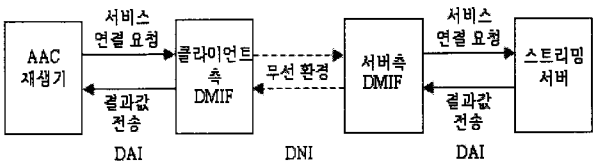


(그림 2) 연산변형을 통한 디코딩 과정

3.2 스트리밍 서비스의 구현

MPEG-4 AAC 데이터의 스트리밍 서비스를 구현하기 위해서, 그림 3과 같은 클라이언트/서버 환경을 구축한다.

서버측 응용프로그램과 재생기 사이의 데이터 전송 및 제어 전달을 위해 MPEG-4의 전달층에 해당하는 ISO/IEC 14496-6 DMIF(Delivery Multimedia Integration Framework)을 이용한다 [9]. DMIF는 두 레벨의 인터페이스인 DAI(DMIF Application Interface)와 DNI(DMIF Network Interface)를 정의하고 있으며, DAI는 DMIF와 응용 프로그램과의 인터페이스를 담당하고, DNI는 DMIF간의 인터페이스를 담당한다.



(그림 3) MPEG-4 AAC 오디오 스트리밍재생 흐름도

스트리밍 서버의 주요 기능은 사용자가 서비스를 요청했을 때 AAC 데이터의 패킷이나 승인되지 못했을 경우는 거부 메시지를 클라이언트에 보내는 것이다. 클라이언트가 요구하는 요청 메시지로는 세션을 설정하는 요청 메시지, 생성된 세션을 해제하는 요청 메시지, 클라이언트의 VCR 연산(열기, 재생, 일시정지, 정지, 종료)을 요구하는 사용자 이벤트 요청 메시지가 있다. 서버는 요청 메시지를 메시지 큐에 저장하고 해석하여 해당하는 연산을 수행한다. 그리고 수행된 결과를 반환하는

응답 메시지로는 세션 설정에 대한 응답 메시지, 세션 해제에 대한 응답 메시지, 클라이언트의 사용자 이벤트에 대한 응답 메시지가 있다. 수행된 결과는 응답 메시지로 메시지 큐에 반환되게 되며 서버측 DMIF를 통해 전송한다.

서버측 DMIF는 스트리밍 서버와의 인터페이스를 정의하고 클라이언트측 DMIF와의 통신을 담당한다. 세부적으로 서버측 DMIF에는 클라이언트측 사용자로부터의 요청을 서버에 전달을 위한 메시지 쓰기 기능, 서버로부터의 응답 메시지를 읽기 위한 메시지 읽기 기능, 서버의 버퍼에 저장된 데이터를 PDA용 MPEG-4 AAC 재생기의 버퍼로 전송하기 위한 전달 기능이 있다. 제어 메시지의 송수신과 데이터의 송신은 각각의 다른 채널을 통해 클라이언트에게 전송한다.

3.3 DMIF간의 인터페이스(DNI)

DMIF간의 인터페이스는 서버측 DMIF와 클라이언트측 DMIF간의 데이터 전송과 제어 전송을 위한 인터페이스를 정의한다. 세부적으로 는 처음으로 서비스의 연결을 위한 세션 연결요청과 서비스의 종료를 위한 세션 종료요청을 수행한다. 또한, 연결된 세션을 통해 데이터 전송을 위한 하나의 트랜스먹스 채널과 사용자의 VCR 연산을 전송하기 위한 제어 채널을 할당한다. 제어 채널은 신뢰성의 보장을 위해 TCP를 이용하며, 트랜스먹스 채널은 신뢰성의 보장은 없으나 빠른 전송효과를 줄 수 있는 UDP를 이용한다.

클라이언트측 DMIF는 클라이언트 응용 프로그램인 PDA용 AAC 재생기와의 인터페이스를 정의하고 서버측 DMIF와의 통신을 담당한다. 세부적으로는 사용자의 요청을 제어 채널을 통해 서버에게 전달하고, 서버로부터의 AAC 데이터를 전송 채널에 의해 정의된 트랜스먹스 채널을 통해 재생기의 버퍼에 저장한다.

4. 구현 및 실험 결과

본 논문에서는 PDA와 같은 낮은 리소스를 가진 시스템에서의 사용을 위해 정의된 LC(Low Complexity) 프로파일을 이용한다. 스트리밍 서버의 구현 환경은 펜티엄급 이상의 CPU와 메모리 64MB이상을 가진 PC이고, 사용된 운영체제는 리눅스(Red Hat Linux 7.3)이다.

제안한 이동단말기용 MPEG-4 AAC 재생기의 구현을 위해 내장형 리눅스 기반의 샤프(Sharp)사의 자우루스(Zaurus) SL-5500을 이용하였다. 운영체제는 내장형 리눅스 커널(Kernel)을 사용하였으며, 탑재된 커널은 Kernel 2.4.6--rmk1-np2-embedix이다. 또한 사용자 그래픽 인터페이스를 위해서는 Trolltech사의 Qt/E and Qtopia를 이용하였으며, 스트리밍 서버와 PDA에서의 통신을 위해서는 무선 랜카드(CF Wireless LAN Card)인 ezLink XI-825을 사용하였다.

성능평가는 연산변형을 통한 MPEG-4 AAC 데이터의 재생 시 실수연산을 통한 디코딩을 수행하였을 때와 오차의 범위를 측정하였다.

표 1에서 보이는 바와 같이 전체 재생시간이 1분 18초(78초)인 'Intro'라는 곡명을 가진 wav 데이터를 MPEG-4 AAC로 인코딩한 다음, 재생기의 각 모듈을 단말기의 환경에 최적화 하지 않은 상태에서의 디코딩 시간을 측정하였다. 이 때 인코딩 옵션은 44100Hz의 샘플링율과 8Kbps 비트율의 LC 프로파일을 이용하였다. 이러한 MPEG-4 AAC 데이터를 1.7GHz의 PC 환경에서 디코딩 시 걸리는 시간은 4.11초인데 반해 PDA에서의 디코딩 시간은 1443.03초로써 약 300배 이상의 시간이 더 소요된다. 즉 원곡의 연산변형이 이루어지지 않은 상태에서의 디코딩 시간이 너무 길어져서 원곡의 재생이 불가능하다.

<표 1> MPEG-4 AAC의 디코딩 속도 비교

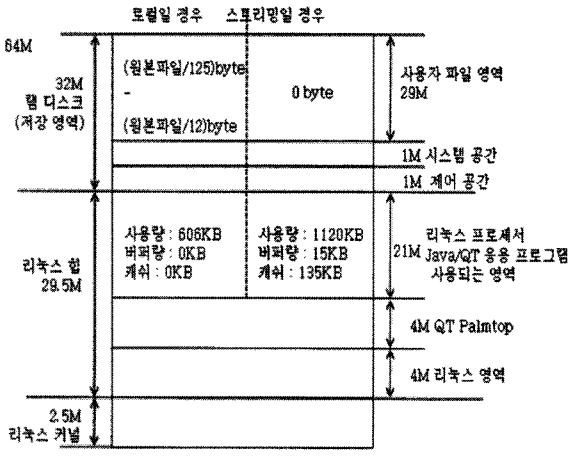
원본 데이터의 정보	MPEG-4 AAC 인코딩 옵션	디코딩 속도	
		PC	PDA
재생 시간 : 1분 18초 Resolution : 16bit 채널 : stereo 샘플링율 : 44100Hz 크기 : 13716 KB	64000bps LC 프로파일	4.65 초	63.35 초
재생 시간 : 1분 18초 Resolution : 8bit 채널 : mono 샘플링율 : 8000Hz 크기 : 622 KB	8000bps LC 프로파일	0.31 초	11.57 초

같은 곡을 인코딩 속성을 달리 하여 MPEG-4 AAC로 인코딩을 행한 다음, 정수형 연산을 통해 디코딩을 행하였을 때의 디코딩 시간을 비교해 보았다. 실험 결과 PDA에서의 디코딩 시간이 재생 시간보다 더 짧아 정상적인 재생이 가능하다.

그림 4는 내장형 리눅스 운영체제가 탑재된 PDA(자우루스 SL-5500)의 경우에 메모리의 전체 구조 및 메모리의 사용량을 보여준다. 우선, 전체 메모리 구조는 크게 3가지 영역인 램 디스크, 리눅스 힙, 리눅스 커널로 나누어지고, MPEG-4 AAC 재생기의 실행 시 필요한 영역은 MPEG-4 AAC 파일을 저장하기 위한 사용자 저장 공간과 MPEG-4 AAC 재생기의 실행 시 생성하는 리눅스 프로세서와 QT 응용 프로그램 실행에 대한 영역이 필요하다.

PDA에 저장된 MPEG-4 AAC 데이터를 재생하기 위해서는 MPEG-4 AAC 파일의 저장공간이 필요하다. 이는 인코딩 시 사용자의 선택에 따라 최대 원본파일의 약 8.3%의 데이터 공간이 필요하다. 하지만, 스트리밍 서비스일 경우에는 서버로부터 데이터를 전송 받기 때문에 어떤 특정한 사용자 파일 영역은 필요치 않는다.

MPEG-4 AAC 재생기를 실행시켰을 때 사용되는 응용 프로그램의 프로세서의 사용량은 로컬일 경우에 606KB만큼을 필요하지만 스트리밍 서비스일 경우의 사용량은 1120KB만큼의 영역이 필요하다.



(그림 4) 스트리밍을 통한 재생 시 메모리 사용량

마지막으로 구현된 재생기의 디코딩 결과값의 오차범위를 조사하였다. 재생시간이 3분 58초의 오디오 데이터를 정수형 연산 변형을 통한 디코딩을 수행했을 때 오차의 분포를 나타내면 다음과 같다. 표 2에서 오차범위는 실수연산을 통한 디코딩 결과값과의 차이를 나타낸다. 100.0 이하의 오차를 가지는 디코딩 결과값이 약 71%이상을 차지하여 실제 재생 시 실수연산을 통한 디코딩 값과의 차이점은 무시해도 좋을 정도로 나타났다.

5. 결 론

MPEG-4 AAC는 높은 음질과 압축률을 제공할 뿐만 아니라 사용자의 선택에 따라 다양한 비트율과 샘플율을 설정할 수 있다. 또한, 시스템의 성능에 맞는 각각의 프로파일을 정의하고 있으며, 이에 따라 인코딩이 가능하다. 특히, 낮은 시스템의 성능에 적용을 위해 LC(Low Complexity) 프로파일을 정의하고 있으며, LC 프로파일로 인코딩된 MPEG-4 AAC는 내장형 시스템에서의 오디오 데이터로써 적합하다.

본 논문은 MPEG-4 AAC 데이터를 PDA 환경에서의 재생기를 구현하고 스트리밍 환경에서의 유효한 데이터로서 활용성을 검증하였다. PDA와 같은 내장형 시스템에서의 MPEG-4 AAC 데이터를 재생하기 위해서 디코딩 연산 변형을 이용한 모듈을 구현하였다. 또한 스트리밍 서비스 환경을 구축하기 위해서 스트리밍 서버 및 전송계층을 통합 구현하였다. 실험 결과를 통해서 하드웨어적 제약조건이 있는 내장형 단말기에서 MPEG-4 AAC 데이터가 효과적으로 재생됨을 보였다. 이러한 오디오 어플리케이션은 내장형 시스템의 이동 단말기에서 다양하게 활용할 수 있을 것이며, 향후 성능향상을 위한 연구가 필요하다.

참 고 문 헌

[1] M. Butrico, N. Cohen, J. Givler, A. Mohindra, A. Purakayastha, and D.G. Shea, "Enterprise Data Access from Mobile Computers:An End-to-End Story," IEEE

Proc. of the 10th International Workshop on Research Issues in Data Engineering, pp.9-16, 2000.
[2] J. Jing, A. Helal, and A. Elmangarmid, "Client-Server Computing in Mobile Environment," *ACM Computing Surveys*, vol.31, no.2, pp.117-157, June, 1999.
[3] ISO/IEC JTC1/SC29/WG11, "Coding of moving pictures and audio- MPEG-2 Advanced Audio Coding AAC," ISO/IEC 13818-7 International Standard, 1997.
[4] http://www.pentium.co.kr/support/performancetools/libraries/ipp/IntelIntegrated Performance Primitives(IPP)
[5] http://www.arm.com
[6] J.C. heudin, C. Panetto, RISC Architectures, CHPMAN & HALL, 1992.
[7] 지화준, 김태훈, 박주성, "AAC 디코더의 IMDCT를 위한 고학 IFFT 알고리즘", *한국음향학회지*, 제26권 제5호 pp.214-219, 2007.
[8] 차경애, "휴대용 임베디드 프로세서에서의 MPEG-4 오디오의 실시간 재생을 위한 정수 디코딩 기법", *방송공학회논문지*, 제 13권 3호, pp.415-418,2008.5.
[9] ISO/IEC 14496-6 "Delivery Multimedia Integration Framework, DMIF," March 1999.

차 경 애 (Kyung-Ae Cha)

- 정회원
- 1996년 2월 : 경북대학교 컴퓨터과학과 (이학사)
- 1999년 2월 : 경북대학교 컴퓨터과학과 (이학석사)
- 2003년 8월 : 경북대학교 컴퓨터과학과 (이학박사)
- 2005년 3월 2007년 3월 : 대구대학교 정보통신공학부 전임강사
- 2007년 4월 ~ 현재 : 대구대학교 정보통신공학부 조교수
- 관심분야 : 멀티미디어시스템, DMB시스템, 멀티미디어 응용