논문 02-27-12C-7 한국통신학회논문지 '02-12 Vol.27 No.12C

MPEG-2 AAC 포맷 기반의 오디오 스트리밍 시스템 설계 및 구현

준회원 이 승 재*, 정회원 이 승 룡**

Design and Implementation of an MPEG-2 AAC Format-based Audio Streaming System

Seung-Jae Lee* *Associate Member*, Sung-Young Lee** *Regular Members*

요 약

현재 주문형 오디오 서비스나 생방송 서비스를 하는 오디오 스트리밍 제품은 제한된 숫자의 클라이언트만을 지원한다는 제약점과, 네트워크 자원의 비효율적인 사용으로 서비스 안정성의 미비와 질적 저하라는 단점을 가지고 있다. 또한, 사용자의 편의성을 위한 동적 서비스 제공에 대한 고려가 미비하여 사용에 불편을 겪는다. 본 논문에서는 이러한 제약점을 해결하기 위한 하나의 방안으로 네트워크 자원을 효율적으로 사용할 수 있는 MPEG-2 AAC 오디오 파일 포맷을 사용하고, 스트리밍 서비스의 안정성과 질적 향상을 위하여 QoS를 지원하는 오디오 스트리밍 전송과 제어에 대한 설계와 구현에 대해 소개한다. 제안된 시스템은 서버와 사용자간의 인터페이스가 고정적인 웹 페이지 하나 만에 의존하고 있어서 정보의 제공에 있어 정적일 수밖에 없는 현재의 정적인 스트리밍 서비스들과는 달리 동적인 서비스를 제공함으로서 사용자와 서버관리에 편의성을 지원하는 새로운 정보관리 기법을 채택하고 있다. 구현 결과 제안된 시스템은 기존의 MP3 파일 포맷을 사용한 스트리밍 시스템 보다 성능이 개선되었고, 서비스의 안정성뿐만 아니라 서버 관리가 용이하다는 장점도 보여주고 있다.

ABSTRACT

Currently, audio streaming services such as on-demand service and live service support only a limited number of clients. They also suffer from a lack of stability and degradation of service quality due to their inefficient use of network resources. Futhermore, since the streaming services usually do not consider dynamic services, they are very inconvenience to use. In order to resolve these drawbacks, we propose a novel audio streaming system based on MPEG-2 AAC file format which are facilitated with the network bandwidths efficiently. The proposed system supports QoS for audio streaming as well as guarantees a stability while streaming service is undergoing. Moreover, the system provides a dynamic interface which enables us to use the streaming service more easily and to manage streaming servers with convenient manner. On the contrary, most of the current available static interface streaming services are mainly depending only on a single fixed web page between client and server, which in consequence lead us to use unflexible static service environment. Our implementation results show the proposed system improves the performance compared to those of the currently existing systems that use MP3 file format. It also provides some benefits such as a stability of service and a easy to management of streaming servers.

* 경희대학교 컴퓨터공학과 실시간 멀티미디어 연구실(crisman@oslab.kyunghee.ac.kr)
** 경희대학교 전자정보학부(sylee@oslab.kyunghee.ac.kr)
논문번호 : 020277-0617, 접수일자 : 2002년 6월 17일

Ⅰ. 서 론

스트리밍 기술은 데이터 전체를 전송한 후 재생하는 것이 아니라 일부 데이터를 수신 후 즉시 재생하는 방식을 사용하는 기술로서 실시간 특성을 가진 멀티미디어 데이터 전송 및 재생에 주로 사용된다. 이런 스트리밍 기술을 사용한 서비스는, 동영상을 기반으로 한 인터넷 방송과 음성, 음악을 전송하는 오디오 방송을 기반으로 한 주문형 서비스가 주류를 이룬다.

현재의 스트리밍 시스템들은 전송 시 데이터 안정성을 고려한 QoS(Quality of Service)기법이 충분히 고려되어 있지 않고, 표준 프로토콜을 사용하지 않아서 다른 스트리밍 시스템들과의 호환성이 떨어지며, 새로운 기능을 추가하거나 새로운 미디어 포맷을 적용할 때 확장성 부족으로 인하여 시스템 수정 시 비용이 많이 소요 된다. 또한, 스트리밍 서버에 미디어를 추가하거나 변경 시 서비스나 정보의 수정과 같은 동적인 환경을 고려하고 있지 않다. 그리고, 복수개의 스트리밍 서버가 존재할 경우 서버 사이의 정보 교환이나 관리가 어려워, 사용자는 각 서버의 정보를 알아야 스트리밍 서비스를 받을 수 있게 된다. 따라서 스트리밍 시스템은 실시간 전송이나 재생 기술 뿐 아니라, 네트워크의 전송 지연과 데이터 손실을 줄이기 위하여 QoS를 지원해야하고, 기존 시스템과의 호환성 및 다양한 환경에 적용할 수 있도록 확장성도 고려되어야 한다. 또한, 복수개의 스트리밍 서버의 원활한 관리와 유동적인 환경을 위한 동적 정보도 제공할 수 있어야 한다.

이러한 점에 착안하여, 본 논문에서는 MPEG-2 AAC(Advanced Audio Coding Standard)포맷을 사용한 개선된 오디오 스트리밍 시스템을 제안한다. 제안된 시스템에서는 실시간 미디어 전송 지원을 위하여 미디어 전송 관리자를 설계하였고, 다양한 미디어 포맷을 지원함으로 확장성을 확보할 수 있는 미디어 관리자를 설계하였으며, 전송 지연과 데이터 손실로 인한 서비스의 안정성 문제를 해결하기 위하여 QoS 기법을 유연하게 적용할 수 있는 QoS 관리자를 설계하였다. 또한, 제안된 시스템은 원활한 스트리밍 시스템의 관리와 구현을 위하여 자원 관리자와 정보 관리자를 두고 있어 복수개의 서버가 존재할 때의 정보제공을 고려하였다. 설계된 시스템을 바탕으로 기존 시스템과의 호환성을 제공하기 위해 RTP, RTSP, SDP등의 표준 프로토콜을 사용하고, 미디어 포맷은 MPEG-2 AAC 오디오를 사용하여 MPEG-2 AAC 오디오 스트리밍 시스템을 구현 한다. 제안한 시스템은 다양한 미디어를 수용할 수 있는 확장성과 QoS를 지원할 수 있도록 설계되어 있기 때문에 향후 차세대 멀티미디어 스트리밍 시스템 개발 시 참고가 될 수 있을 것으로 예측된다.

논문의 구성은 다음과 같다. Ⅱ장에서 오디오 스트리밍 기술의 관련 연구를 살펴보고, Ⅲ장에서는 오디오 스트리밍 시스템의 설계 모델을 소개하고, Ⅳ장에서는 제안된 오디오 스트리밍 시스템의 설계 내용을 살펴본다. Ⅴ장에서는 설계한 시스템을 바탕으로 시스템 구현 내용에 대하여 기술하고, Ⅵ장에서는 구현된 시스템의 성능 평가를 논의한 뒤, Ⅶ장에서 결론을 내린다.

Ⅱ. 관련 연구

1. 오디오 스트리밍 연구

Null-soft의 Winamp를 기반으로 한 오디오 스트리밍에서 Winamp는 현재 많은 사용자가 이용하고 있는 오디오 플레이를 위한 Nullsoft사의 프리웨어 소프트웨어로서 이를 기반으로 Shout-cast 스트리밍 시스템이 개발되어 있다[15]. 지원되는 오디오 데이터 포맷으로는 MP3, VQF, WAV 등이 있다. 이런 오디오 포맷들은 Plug-in 형식으로 지원되고 있으며 TCP/IP와 HTTP 기반으로 동작하고 있다. 그러나, 이 스트리밍 시스템은 기술적으로 다음과 같은 문제점을 지니고 있다. 첫째, 신뢰성을 기반으로 한 연결지향적인 TCP 프로토콜 전송과 HTTP 프로토콜은 모든 작업이 끝날 때까지 네트워크 자원을 점유하기 때문에 스트리밍 기술에 적합하지 않다. 둘째, 스트리밍 서비스의 안정성을 위한 QoS기법이 고려되어 있지 않아서 연결이 끊기거나 데이터의 전송이 지연되는 등의 네트워크에서 문제가 발생하면 서비스가 진행되지 않거나 음질의 저하를 가져오게 된다. 이에 반하여 본 논문에 서 제안하고 있는 시스템은 UDP와 TCP 프로토콜 모두를 지원하고, 네트워크로 인한 에러를 최소화하기 위하여 QoS 기법인 재전송과 FEC방법을 고려하고 있다.

WMT(Windows-Media-Technologies)는 WMA(Windows Media Audio) 포맷을 중심으로 다양한 오디오 포맷을 사용하고 있는 Microsoft사의 스트리밍 시스템이다. 이는 MMS라는 MS고유의 세션 제어 프로토콜을 사용하고 있으며, UDP/IP,

TCP/IP, RTP, HTTP등과 같은 다양한 네트워크 전송 프로토콜을 이용하여 데이터의 전송을 할 수 있다[2]. 하지만, WMA 오디오 포맷은 국제적인 표준이 아니며, 64Kbps이상에서만 음질의 비교우위를 가진다. 또한, HTTP기반의 스트리밍이 주를 이루기 때문에 네트워크의 대역폭을 많이 사용하는 단점을 가지고 있다. 이에 반하여 제안된 시스템은 이런 WMT에서 미약한 QoS 부분을 개선하고 비표준 사용으로 인한 확장성의 결여를 해결하기 위하여 표준 프로토콜을 사용한 시스템을 설계하고 있다.

Real-System 스트리밍 시스템은 고유의 미디어 포맷(.rm, .ram 등)과 AVI, WAV 등의 다양한 오디오 포맷을 지원하며 현재 다수의 인터넷 방송에서 활용중인 스트리밍 시스템 중 하나이다[8]. TCP/IP기반의 HTTP스트리밍을 지원하고 RTP 기술도 사용되고 있다. 또한, 세션을 제어하는 기술로서 RTSP 프로토콜을 사용하고 있다. Real-System 스트리밍 시스템은 다양한 네트워크 환경의 지원과 압축률이 높은 멀티미디어 데이터 포맷을 이용한 스트리밍 서비스를 사용하기 때문에 동시에 많은 사용자들에게 양질의 서비스를 제공할 수 있다. Real-System의 스트리밍 시스템의 다양한 미디어 확장성을 보완하기 위하여 본 논문에서 제안하는 시스템은 오디오 표준, MPEG-2 AAC 오디오 등 다양한 표준 오디오를 지원하기 위한 확장성을 고려하고 있다.

2. MPEG-2 AAC 오디오 포맷

MPEG-2 AAC 오디오는 낮은 Bit 등급에서 고음질을 실현 할 수 있도록 표준화한 MPEG-1과 호환성이 없는 오디오 부호화 방식이다. 기존의 MP3는 곡 전체의 정보를 담는 헤더 뒤에 데이터가 프레임 단위로 저장된다. 프레임의 크기는 헤더의 정보에 포함되어있으면 고정된 프레임 단위로 압축을 하게 된다. 이 프레임 구조는 압축률이 높은 부분에서도 고정적이기 때문에 불필요한 용량을 차지하여 오디오 데이터의 용량을 크게 만드는 요인으로 작용되고 있다. 하지만 AAC는 프레임의 구조가 가변적으로 되어 있어서 압축률에 따라 그 크기가 변하기 때문에 불필요한 부분을 제거할 수 있어서 전체 오디오 파일의 용량이 줄어들게 된다. 실제로 MP3파일과 비교했을 때 30% 이상의 압축률을 보이고 있다. 32Mbyte의 저장 공간인 경우 4분 정도의 음악을 MP3로는 8곡을 담을 수 있지만 AAC는 11곡 이상을 담을 수 있다. MP3와는 다른 유동적인 데이터 구조를 가진 AAC 포맷의 저용량 특징은 네트워크의 부하나 전송지연으로부터 스트리밍의 안정성을 높일 수 있는 장점으로 작용하고 있다.

3. 오디오 스트리밍의 QoS 기술

오디오 스트리밍에서 쓰이는 데이터는 연속성에 민감하여 적은 데이터의 손실에도 서비스에 대한 신뢰성에 크게 영향을 미칠 수 있다. 네트워크를 통하여 실시간적인 전송을 하고, 수신한 데이터를 플레이하는 스트리밍의 경우 네트워크의 부하에 따른 전송 지연 및 데이터의 손실이 나타날 확률이 높다[9]. 이런 스트리밍의 특징 때문에 스트리밍은 실시간적인 속성을 잃어버릴 수 있고, 사용자에게 불편함을 줄 수 있다.

FEC(Forward-error-Correction)방법은 Best-effort service상에서의 QoS를 위해 손실된 패킷을 처리하는 방법으로 첫 번째 이후의 패킷에 이전 패킷 정보인 lower quality data를 포함하게 하여 패킷 손실이 있었을 때 이를 수정하는 방법이다. 이 방법은 단순히 인접한 두 패킷만을 모니터링 하면 되므로 Play-out 지연시간이 줄어들게 된다. 또한, 전송 패킷에 포함되는 부가 정보는 원본 데이터가 아닌 lower quality data로서 용량이 작으므로 전송시의 부하를 줄일 수 있고, 전송이 실패했을 경우에도 재생의 지속적인 수행이 가능하게 된다. 하지만 원본 데이터보다는 음질이 떨어지는 단점이 있다[3].

실시간 오디오 스트리밍에서 전송 표준으로 사용하고 UDP(User Diagram Protocol) 표준은 많은 장점을 가지고 있음에도 불구하고 데이터의 손실을 가져올 수 있는 단점을 지니고 있다. 이런 UDP 기반의 데이터 전송으로는 데이터의 손실에 대한 신뢰성을 확보할 수가 없고, 네트워크 지연으로 인한 패킷의 손실을 막을 수 없다. 이를 보정하는 방법으로 손실된 패킷을 다시 전송하는 것이 재전송 방법이다. 이런 재전송 방법은 FEC의 방법에서 문제점으로 지적되는 음질 저하 문제를 해결할 수 있지만, 네트워크의 자원을 많이 사용한다는 단점을 가진다[9].

Ⅲ. MPEG-2 AAC 오디오 스트리밍 시스템 모델

본 논문에서 제안하는 오디오 스트리밍의 모델은 그림 1과 같으며 이 모델은 MP3와 AAC 등의 오디오뿐만이 아닌 다양한 미디어 타입을 모두 적용할 수 있는 미디어 확장성을 가지고 있고, QoS의

적용이 용이한 유연한 구조를 지닌 스트리밍 시스템 모델이다. 이 오디오 스트리밍 시스템은 오디오 스트리밍 어플리케이션을 위한 어플리케이션 인터페이스 아키텍처(Architecture)와 오디오 스트리밍의 전송 및 제어와 관련한 환경 아키텍처(Environment Architecture)로 구성된다.

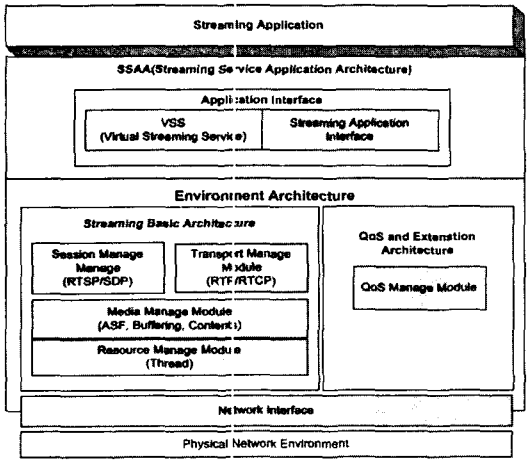

그림 1. MPEG-2 AAC 스트리밍 시스템의 모델 구조

Environment Architecture는 Transport Manage Module과 Session Manage Module, Media Manage Module, Resource Manage Module, Qos Manage Module로 구성된다. 오디오 데이터의 전송을 위한 관리 모듈(Transport Manage Module)은 오디오 데이터를 네트워크로 전송하기위해 알맞게 분해, 조립하는 기능과 패킷화 된 오디오 데이터를 실시간으로 전송하는 RTP 표준 프로토콜을 이용하여 설계하였다[5]. 오디오 스트리밍의 제어를 위한 관리 모듈(Session Manage Module)은 미디어 채널의 생성과 제어를 담당하는 기능을 가지고 있으며, RTSP 표준 프로토콜을 사용하여 설계하였다[6]. 이 세션 제어 모듈은 메시지 처리를 주로 하는 모듈로서 SDP 프로토콜을 이용하여 미디어 정보를 제공하고 처리한다. 이 전송 기술과 세션 제어 기술은 네트워크 인터페이스를 통한 송수신 기능을 수행함으로써 다양한 네트워크 프로그래밍 인터페이스를 지원할 수 있는 유연한 구조를 지니고 있다. 이 전송 모듈과 제어 모듈은 미디어 정보를 저장하고 다양한 미디어를 스트리밍 할 수 있도록 확장성을 위해 만든 미디어 관리 모듈(Media Manage Module)과 프로그램의 원활한 동작을 위한 자원 관리 모듈(Resource Manage Module)을 하위에 포함하고 있다. QoS 관리 모듈(QoS Manage Module)은 FEC와 재전송 등의 QoS 기법의 적용을 위한 정책 결정 및 적용 모듈이다.

스트리밍 서비스를 위한 어플리케이션 인터페이스 Architecture는 스트리밍 서버와 클라이언트와 같은 스트리밍 어플리케이션을 위한 어플리케이션 인터페이스와 VSS로 불리는 정보 관리 모듈로 구성된다. 이 VSS 모듈은 스트리밍 서버의 미디어 정보와 복수개의 서버 정보를 동적으로 관리하여, 스트리밍 서버의 투명성과 DNS(Domain Name Service) 개념에 익숙한 사용자에게 편의성을 제공을 하도록 설계하였다. VSS Module은 스트리밍 서버와 클라이언트간의 정보를 공유하기 위한 목적으로 만들어진 것으로 세션 관리 모듈과 비슷한 기능을 한다. 마지막으로 네트워크 인터페이스 모듈(Network Interface Module)은 TCP, UDP, 멀티캐스트, 유니캐스트, 브로드캐스트 등의 다양한 미디어 전송환경을 고려하여 표준 네트워크 프로토콜을 참고하여 만든 전송 하위 모듈이다. 이 모듈을 사용하여 전송과 제어의 통신을 하게 된다. 그림 2는 MPEG-2 AAC 스트리밍 시스템의 모델을 구조이다.

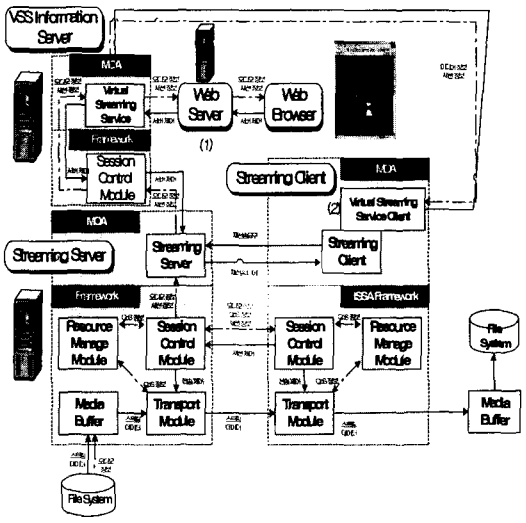

그림 2. MPEG-2 AAC 스트리밍 시스템의 논리적 구조

Ⅳ. MPEG-2 AAC 오디오 스트리밍 시스템 설계

1. 네트워크 인터페이스(Network Interface) 설계

네트워크 인터페이스는 그림 3과 같은 구조를 가

지며 네트워크 환경에 독립적으로 실행될 수 있고 다양한 네트워크 환경을 지원하는 일종의 네트워크 wrapper 인터페이스이다. 현재는 윈도우 환경의 Win-Sock 및 유닉스 환경의 버클리 소켓에 대한 인터페이스를 제공하며, TLI(Transport Layer Interface), ATM(Asynchronous Transfer Mode) 등의 다양한 네트워크 프로그래밍 인터페이스를 지원할 수 있는 유연한 구조를 지니고 있다. 또한, 네트워크 인터페이스를 통해 유니캐스트와 멀티캐스트를 손쉬운 적용이 가능하며, 스트리밍 시스템의 전송 관리 모듈이나 세션 관리 모듈과 같이 네트워크 사용이 많은 다른 모듈에서 이용되어 프로토콜의 독립성을 지원 해준다. 이동 통신 환경 등의 다양한 물리적 네트워크 환경을 고려한 확장성 있는 구조로 되어있다.

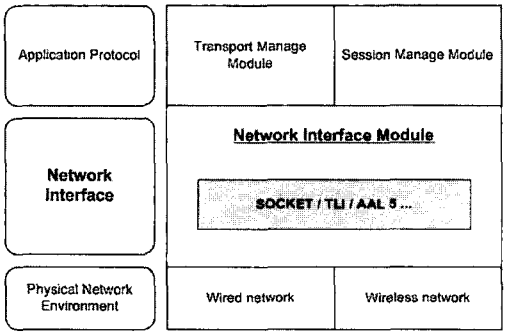

그림 3. 네트워크 투명성 제공 및 확장성을 고려한 네트워크 인터페이스

2. 전송 관리 모듈(Transport Manage Module) 설계

스트리밍 데이터 전송 관리 모듈은 스트리밍 시스템에서 오디오 데이터를 알맞게 분해, 조립하는 기능과, 패킷화 된 미디어 데이터를 네트워크로 실시간 전달하는 역할을 수행한다. 즉, 전송 관리 모듈은 네트워크 환경에서 오디오 데이터를 스트리밍 하기 위한 유연성 있고, 확장성 있는 전송 환경을 제공한다. 그림 4는 전송 관리 모듈의 전송 프로토콜 스택으로서 전송 관리 모듈에서는 RTP 기반의 오디오 전송 프로토콜과 IP-유니캐스트 및 IP-멀티캐스트 동작환경을 모두 지원한다. 또한, 재전송을 위해 신뢰성 기반 TCP를 전송 관리 모듈에서 지원하고 있다.

전송 관리 모듈은 제안된 시스템에서 전송되는 데이터를 알맞게 분해, 조립하는 기능과, 패킷화 된 미디어 데이터를 다양한 네트워크 환경에서 실시간 전달하는 역할을 수행한다. 즉, 전송 관리 모듈은 다양한 네트워크 환경에서 멀티미디어 데이터를 스트리밍 하기 위한 유연하고 확장성 있는 전송 환경을 제공한다. 전송 관리 모듈에서는 RTP, TCP기반의 전송 프로토콜과 IP-유니캐스트 및 IP-멀티캐스트, 이동 통신 등의 동작환경을 모두 지원한다. 또한, 이동 통신 네트워크의 대역폭이나 기타 특징적인 환경에 맞게 패킷 사이즈와 전송 간격을 변화시킬 수 있도록 하고 있다.

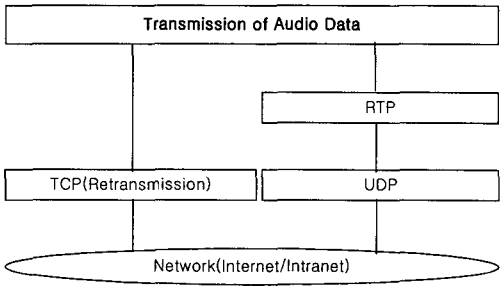

그림 4. 전송 관리 모듈의 전송프로토콜 스택

3. 세션 관리 모듈(Session Manage Module)의 설계

제안한 시스템의 전송 관리 모듈에서 사용하고 있는 RTCP 표준은 최소한의 정보만을 모니터링 할 수 있다. 그러나, 이는 스트리밍 제어로서는 그 기능이 미약하다. 따라서, 제안된 시스템에서는 스트리밍을 보다 원활하고 강력하게 제어하기 위하여 RTSP를 이용한 스트리밍 세션 관리 모듈을 설계하고 구현하였다. 세션 관리 모듈은 오디오 스트리밍 응용 프로그램 사이의 연결에 관한 모든 것을 생성하거나 소멸, 재생, 멈춤 등의 기능을 제공함으로써 스트리밍 되는 오디오를 제어 할 목적으로 설계되었다. 세션 관리 모듈에는 추가로 SCP (Session-Control-Protocol)를 지원하고 있으며, SCP는 멀티캐스트 채널 관리 기능과 다양한 스트리밍 정보를 효율적으로 분배 및 관리하고, 스트리밍 서버를 관리할 목적으로 RTSP 표준 프로토콜을 참고하여 설계되었다. 그리고, 세션의 정보를 기술하기 위한 표준 프로토콜인 SDP를 사용하여 세션을 기술함에 있어 표준화에 따른 범용성을 확보하였다. SDP에 담긴 정보는 스트리밍 서버와 클라이언트 사이에서 RTSP를 통해 전송되고, 스트리밍 서버와 웹 서버 사이에서는 SCP를 통해 전송되어 사용자나 관리자에게 현재 스트리밍 서버의 상태를 알려주는데 사용되고 있다. 또한, 세션 관리 모듈은 세션 성립 이전과 성립 이후 관리에서도 편리성을 확보하기 위

하여 여러 부가적인 프로토콜을 지원하고 있다.

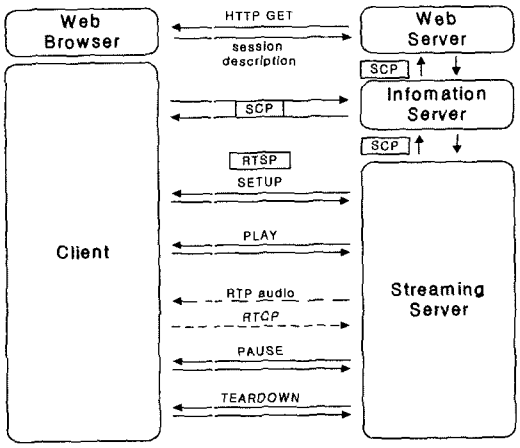

그림 5. RTSP를 사용한 스트리밍의 과정

그림 5에서는 RTSP를 이용한 스트리밍 과정을 보여준다. RTSP는 HTTP나 여타 다른 방법을 사용하여 세션 기술(description)이라는 현재의 세션 정보를 먼저 얻어 오며, 획득된 정보와 클라이언트의 네트워크 정보, 획득하려는 미디어 정보를 SETUP 과정을 통해 서버와 협상을 거친 후에 PLAY에 의해 스트리밍이 수행되며, 이때 PLAY, PAUSE를 통해 VCR과 비슷한 미디어 제어를 할 수가 있다. TEARDOWN 메소드는 모든 수행을 마친 후에 세션을 정리할 때 사용된다. 현재 스트리밍 기술 전반에서 세션 기술 프로토콜로서 사용되고 있다. SDP는 RTSP와 마찬가지로 텍스트 기반의 프로토콜이며, 네트워크나 전송 프로토콜 독립적으로 작동되며, 일반적으로 RTSP와 쌍으로 사용된다. 그림 6은 RTSP 메시지의 구조로서 정보 관리 모듈의 SCP의 메시지 구조와 동일하다.

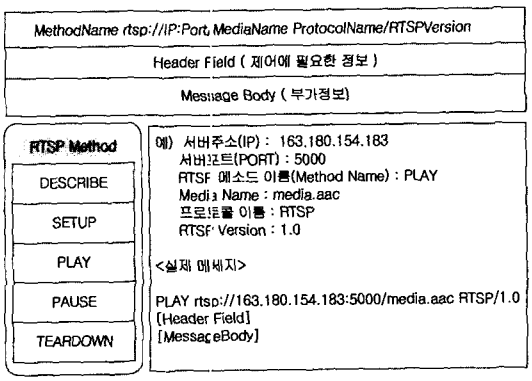

그림 6. RTSP 메시지 구조

4. 미디어 관리 모듈(Media Manage Module)과 자원 관리 모듈(Resource Manage Module)설계

스트리밍 시스템은 하나의 미디어 포맷이 아닌 다양한 포맷을 지원하는 확장성을 가지고 있어야 한다. 또한, 스트리밍의 제어와 전송을 위해서 미디어의 정보를 관리해야 한다. 이를 위해서 본 논문에서는 다양한 미디어의 확장성을 고려하고, 미디어 정보 관리의 편의성을 제공하기 위한 미디어 관리 모듈을 설계하였다. 이 모듈은 SDP 표준 프로토콜을 기본으로 하여 정보를 관리하고, 제공하며, 미디어 정보가 필요한 세션 관리 모듈과 전송 관리 모듈의 하부 모듈로서 동작을 하는 구조이다. 자원 관리 모듈은 스트리밍 시스템 어플리케이션에서의 전송 관리 모듈과 세션 관리 모듈의 원활한 동작을 위한 스레드 및 파일 동시성 제어와 관련한 모듈이다. 이 두 모듈은 스트리밍 시스템의 원활한 동작을 위해 필요한 모듈로서 다른 시스템 및 어플리케이션에서 쓰일 수 있는 독립적인 구조로 모듈의 재사용성이 높도록 설계하였다.

5. QoS 관리 모듈(QoS Manage Module)설계

FEC와 재전송 지원을 위한 전송 모듈은 기존의 RTP/RTCP 전송 관리 모듈에 FEC 패킷과 재전송 패킷과 관련한 모듈인 QoS 관리 모듈과 통합된 형태로 설계되었다. 그림 7은 재전송 기법과 FEC 기법을 사용한 QoS 관리 모듈과 전송 관리 모듈의 통합 구조를 나타낸다. 본 논문에서 제안하는 시스템은 FEC와 재전송 방법의 장점, 단점에 맞는 시스템을 구현할 수 있는 QoS 지원 방법을 포함하고 있다. QoS 관리 모듈에서는 각 네트워크 환경이나 미디어의 종류에 따라 QoS 지원 방법을 선택할 수 있는 정책에 관한 내용을 포함하고 있다. FEC와 재전송 기법을 고려한 QoS 관리 모듈과 전송 관리 모듈의 통합 구조는 그림 7에서 보는 바와 같이 크게 RTP 패킷을 조립하고 분해하는 RTP Packet Builder, RTP Packet Parser, 재전송을 담당하는 재전송 서버 모듈과 네트워크를 통하여 패킷을 보내는 네트워크 전송 모듈로 구분된다. RTP 패킷을 조립하는 RTP Packet Builder는 기본적인 RTP 패킷을 만들 수 있는 모듈을 가지고 있으며, FEC 기법을 사용할 경우를 고려하여 FEC 패킷 Builder를 설계, 구현하였다. FEC 패킷이라고 이름 붙인 이 패

킷은 low quality data를 포함한 RTP 패킷이며 이 RTP 패킷은 클라이언트로 전송되어진 후 클라이언트에서 FEC Packet Parser로 분해 된다.

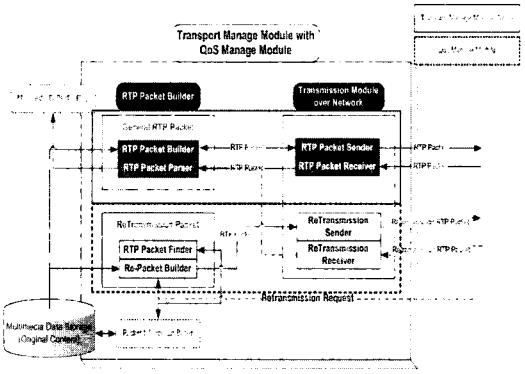

그림 7. FEC와 재전송 기법을 고려한 QoS 관리 모듈과 전송 관리 모듈

6. 스트리밍 시스템의 정보 및 미디어 관리 모듈(VSS) 설계

현재 대부분의 오디오 스트리밍 시스템들은 웹서버와 연동한 단순한 링크 수준이거나 스트리밍 서버가 제공하는 정보에 의한 스트리밍 서비스이다. 데이터 전송에 기인한 서버와 네트워크 자원 부하가 많은 스트리밍 시스템에는 이 같은 점이 응답지연, 네트워크 혼잡, 서버의 부하를 야기 시킨다. 또한, 복수개의 서버인 경우 정보관리 및 미디어 정보 제공이 용이하지 않았다. 복수개의 서버와 미디 정보의 원활한 관리를 위해 스트리밍 서버 정보를 관리하는 모듈을 설계하였고 이를 스트리밍 시스템에 적용함으로써 사용자에게 투명성을 제공한다. 이 같은 정보관리 및 서비스 메커니즘은 다수의 스트리밍 서버를 클라이언트에게 하나의 서버로 보이게 할 수 있으며, QoS를 위한 서버의 로드밸런싱을 가능케 한다. 즉, 정보 관리 모듈의 목적은 복수의 스트리밍 서버에서 컨텐츠와 컨텐츠 정보 공유, 그리고 현재 서비스되고 있는 채널(세션)에 대한 제어와 정보 제공을 목적으로 한다. 정보 관리 모듈은 크게 스트리밍 서버에 포함되는 서버모듈과 클라이언트 모듈이 있으며 모든 작동은 네트워크를 통한 메시지 통신에 기반하고 있다. 이 메시지를 처리하는 것은 정보 관리 서버가 담당하며 이는 스트리밍 서버와 별개로 동작하는 정보 관리 서버이다. 정보 관리 기법은 '가상적인 하나의 스트리밍 시스템(Virtual Singly Streaming System)'을 제공하므로 VSS라고 칭하였다.

V. 오디오 스트리밍 시스템 구현

1. 전송 관리 모듈의 구현

전송 관리 모듈의 구현은 윈도우와 유닉스 환경에서 모두 동작할 수 있도록 ANSI C++로 구현하였으며, 플랫폼 의존적인 코드는 다양한 플랫폼으로 이식 가능한 쓰레드 타이머, 네트워크 인터페이스 라이브러리를 만들어 사용함으로써 쉽게 구현이 가능하게 하였다. 네트워크 인터페이스는 윈도우 환경의 Win-Sock과 유닉스 환경의 버클리 소켓을 지원하여 하나의 일관된 인터페이스로 네트워크에 대한 접근할 수 있도록 한다. 타이머의 경우 패킷을 주기적으로 전송하기 위해 필요한 기법으로 미디어 데이터 패킷이나 혹은 제어 패킷이 서버 측으로부터 클라이언트 측으로 Push하는 방식을 이용한다. 전송 관리 모듈은 그림 8과 같이 논리적으로 크게 전송 세션, 전송 데이터 그리고 전송 제어의 세 부분으로 구성된다. 전송 세션 부분은 전송 세션의 상태, 송수신 통계 등의 전송 세션에 관한 모든 정보를 저장하고, 전송 데이터는 데이터 패킷의 송수신 기능을 수행하며, 마지막으로 전송 제어는 제어 패킷의 송수신 기능을 수행한다. 전송 데이터는 RTP 프로토콜을 기반으로 구현되었고, 전송 제어는 RTCP 프로토콜을 기반으로 하여 구현되었다. 전송 관리 모듈은 QoS 관리 모듈과 통합될 수 있는 확장형 구조를 가지고 있다.

2. 세션 관리 모듈의 구현

세션 관리 모듈은 크게 인터페이스, 메시지, 세션 제어 클래스 모듈로 나누어진다. 인터페이스 클래스 모듈(Interface Class Module)은 세션 관리 모듈 상위의 응용 계층에게 일관성 있는 단일 인터페이스와 원하는 서비스를 조합하여 사용할 수 있는 서버 클래스나 클라이언트 클래스를 제공 해준다. 응용 계층에서는 이러한 인터페이스 클래스들을 이용하여 복수의 서비스를 통합할 수 있다. 메시지 클래스 모듈(Message Class Module)은 인터페이스 클래스에서 사용되며 메시지를 생성, 해석하는 기능을 한다. 세션 클래스 모듈(Session Class Module)은 복수의 세션을 관리하며, 인터페이스 클래스 집합에 의해서 내용이 변경, 추가, 삭제된다. 제어 클래스 모듈(Control Class Module)은 클라이언트의 요청을 받아들이는 스케줄링 기능을 담당하며 이 부분에는 승인과 같은 보안에 관련된 기능을 추가할 수 있도

록 확장을 고려하였다.

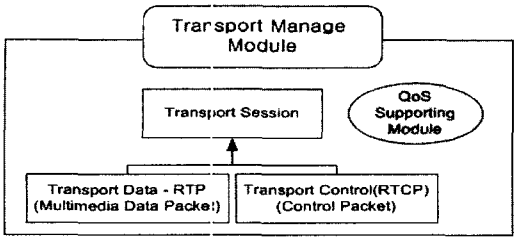

그림 8. 전송 관리 모듈의 구성

3. 인터페이스 클래스 모듈의 구현

인터페이스 클래스는 세션 관리 모듈이 지원하는 프로토콜로 구성되며, 서버 인터페이스 클래스와 클라이언트 인터페이스 클래스로 나누어진다. 서버 인터페이스의 경우 실행(Run) 메소드를 통해 초기화 과정이 수행되고, 간단한 송신/요청 메소드의 호출로 메시지를 송수신 할 수 있다. 클라이언트 인터페이스의 경우 RTSP를 통하여 자동적으로 서버와 연결을 이루며 UDP 연결도 통일된 인터페이스를 위해 TCP 연결과 같은 방법으로 동작한다. 이러한 네트워크를 사용하는 세션 관리 모듈은 네트워크 인터페이스를 사용함으로써 프로토콜의 독립성을 보장받을 수 있다.

그림 9에서 보듯이 서버 인터페이스와 클라이언트 인터페이스의 요청, 응답에 관한 처리 과정은 서버 인터페이스의 Run()함수에 의해 클라이언트의 접속 요청을 받는 쓰레드가 동작하면서 시작된다. AcceptThread의 역할은 클라이언트의 요청을 받기 위한 것이며, 그 후 받아들인 요청의 처리를 위해 작업 쓰레드를 생성한다. 작업 쓰레드인 Request Thread는 클라이언트와 서버사이의 실제 메시지를 주고받는다.

이와 같은 작업이 끝났을 때 서버인터페이스는 AcceptThread에게 RequestThread의 작업이 끝났음을 통지하고, 클라이언트 인터페이스의 경우 DeinitChannel() 함수를 통해 연결을 종료하는 작업을 하게 된다. 한편, 서버 인터페이스 클래스를 상속받은 RTSP, SCP 클래스들은 해당 프로토콜의 메소드 처리를 위한 기능만을 가지고 있으며, 상위 클래스인 smServerInterface에서 메시지 처리를 위한 메시지 핸들링 기능을 가지고 있다. 클라이언트 인터페이스는 서버 인터페이스와는 달리 프로토콜마다 자신의 메소드를 지원하는 함수를 가지고 있다.

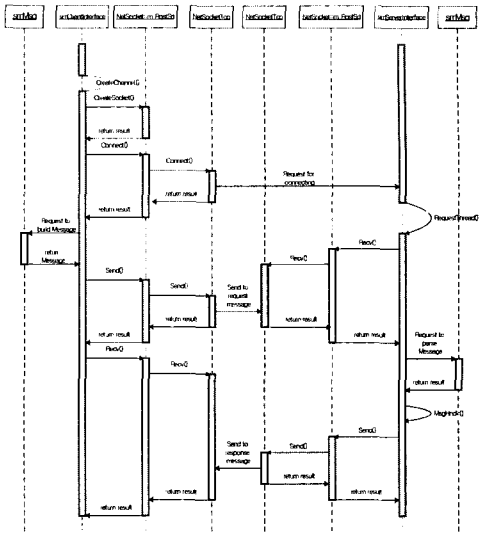

그림 9. 클라이언트와 서버 사이의 메시지 전송 및 처리 시퀀스 다이어그램

4. 메시지 클래스 모듈의 구현

메시지 클래스 모듈은 크게 RTSP, SCP, SDP 메시지로 나누어진다. 각 메시지 클래스 모듈은 파서 메소드와 빌드 메소드를 가지고 있으며, 파서 메소드는 네트워크에서 보내진 패킷을 해석하여 클래스 안에 정보를 저장하는 것이며, 빌드 메소드는 현 클래스내의 정보를 패킷으로 만드는 역할을 한다. URI와 프로토콜 이름, 프로토콜 버전의 정보는 메시지 클래스라는 부모 클래스에 두어 이를 상속받은 자식 클래스에서 사용하며, 헤더 정보 또한 Linked-List를 이용하여 저장해둔다. 요청 클래스의 경우 메소드 이름을 변경하여 사용하며, 응답 클래스의 경우 상태 코드와 Reason Phrase 때문에 세션 상태 클래스를 사용하고 있다. smMsg 클래스는 메시지 클래스의 최상위 클래스로서 추상 클래스로 작성되어 있다. BuildMsg(), ParseMsg()라는 순가상 함수를 하위 클래스에서 구현하도록 권장하고 있다. smRtspMsg, smScpMsg, smSdpMsg는 프로토콜의 메시지를 나타내며, 상태 코드를 smStatusCode, 각각 클래스에 헤더 필드들을 정의하여 사용하고 있다. 그리고, smFactory로부터 메시지의 인스턴스를 받아 사용한다.

그림 10은 서버 인터페이스와 클라이언트 인터페이스에서 메시지를 생성하는 과정을 보여 준다. smSession 객체로부터 세션의 정보를 얻어온 후에 이 정보를 사용하여 요청 라인이나 상태 라인을 생성한 후에 BuildHeaderFld() 함수를 통해 헤더 필드

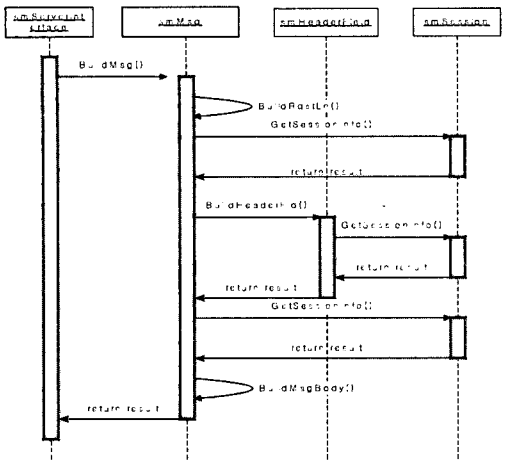

그림 10. 메시지 생성 시퀀스 다이어그램

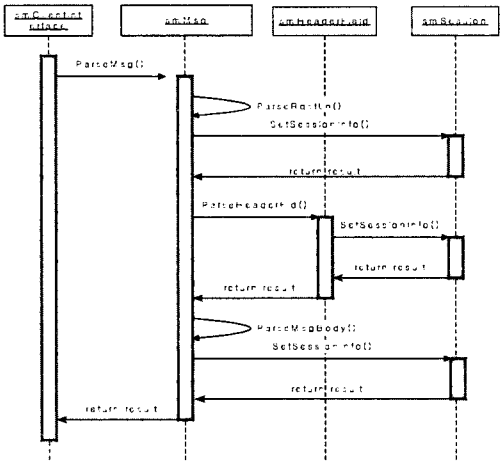

그림 11. 메시지 해석 시퀀스 다이어그램

를 생성하게 된다. 메시지 바디가 있을 경우 BuildMsgBody() 함수를 통해 메시지 바디를 생성한다. 그림 11은 서버 인터페이스와 클라이언트 인터페이스에서 메시지를 해석할 때의 시퀀스 다이어그램이다. 메시지를 생성하는 과정과 비슷하지만, 메시지를 해석한 결과를 저장하는 것과 메시지 바디의 처리 부분이 메시지 생성 경우와 다르다.

5. QoS 관리 모듈의 구현

QoS 관리 모듈은 QoS를 위한 전송 모듈과 통합되어서 동작 되도록 구현되었다. FEC와 재전송, TCP와 UDP, 유니캐스트와 멀티캐스트 등의 다양한 환경 및 요소를 인자값으로 두어 QoS 정책을 다양한 환경에 맞도록 구현하였다. 이 QoS 관리 모듈을 실제로 스트리밍 시스템이 동작할 때 패킷의 손실이 없으면 FEC 패킷의 low quality data 는 버리고 패킷의 손실이 있는 경우에는 low quality data를 이용하여 재생하게 된다. 재전송 모듈은 재전송의 요청을 받아들이는 것으로 패킷을 찾는 모듈과 재전송 패킷 빌더, 재전송 패킷 송신 모듈로 구성된다. 세부적으로 패킷 Finding 모듈은 재전송 요청된 패킷을 정보 버퍼를 사용하여 파일에서 데이터를 읽어오는 역할을 하고, 재전송 패킷 빌더는 재전송 요구된 데이터를 RTP 패킷으로 만드는 역할을 한다. 재전송 패킷 송신 모듈은 재전송 할 RTP 패킷을 신뢰성을 위해 TCP 프로토콜을 사용하여 전송하는 기능을 한다. 한편, 재전송된 데이터는 송신한 것과 마찬가지로 TCP 프로토콜을 사용하여 수신하며, 수신된 패킷은 분해 되어 파일이나 버퍼에 미디어 재생을 위해 보관된다.

그림 12는 QoS 모듈의 동작 알고리즘으로 클라이언트의 요청에 의해 전송 모듈이 동작될 때 FEC 패킷 모드인지 일반적인 RTP 패킷 모드인지를 구별하여 패킷을 만들어 전송하는 것을 보여준다. QoS 관리 모듈은 QoS를 위한 방법이 FEC Flags 인자를 통하여 FEC인지 재전송인지 파악을 한다 (line 1). FEC전송이 아니면 일반 패킷을 전송하는 관리 모듈을 동작시키고(line 3), 재전송 서버를 모듈을 동작시킨 후 요청한 클라이언트로 미디어의 스트림을 보낸다(line 4, line 5, line 6). FEC 패킷

```
QoS Module( Re-transmission, FEC Packet )
line 1 If there are user's streaming requirement
line 2          IF FEC flags = 0   then {
line 3          General Packet Transport Manager Initialize;
line 4          Re-transmission Server Initialize;
line 5          Re-transmission Server Start;
line 6     Server send a media stream to the Client; }   ELSE   then {
line 7          FEC Packet Transport Manager Initialize;
line 8          FEC Packet generator Start;
line 9          Server send a media stream to the Client;   }

Re-Transmission
line 10 If there are user's Packet loss requirement
<SERVER SIDE >
line 11    IF FEC flags = 0    then {
line 12   send a requirement to the Re-transmission Server module;
line 13    Packet Building in Re-transmission Packet builder;
line 14    Server send a Re-transmission Packet to the Client;
line 15    Re-transmission Server Waiting;          }

FEC
line 16 If there are user's Packet loss happen
< CLIENT SIDE >
line 17          send a requirement to the FEC module;
line 18          Searching ing FEC Packet;
line 19          lost Low Data Packet insert to Buffer;
```

그림 12. QoS 모듈의 동작 알고리즘

전송이면 FEC 패킷 관리 모듈을 동작시키고 클라이언트로 미디어의 스트림을 보낸다(line 7, line 8, line 9). 패킷 손실이 일어났을 경우 일반적인 패킷이면, 서버 전송 모듈에 정보를 전송한 후, 패킷 빌딩 모듈에서 재전송 패킷을 만들어 전송을 하게 된다(line 10, line 11, line 12, line 13, line 14, line 15). FEC 패킷이면, 서버로의 정보 전송이 필요 없이 클라이언트의 FEC 패킷 관리 모듈에 패킷 손실 정보를 보내고 패킷에서 낮은 데이터율의 부분을 찾아 미디어 재생을 위한 버퍼에 넣는다(line 16, line 17, line 18, line 19).

6. 스트리밍 시스템의 정보 및 미디어 관리 모듈(VSS) 구현

정보 관리 모듈은 정보를 미디어의 정보 및 서버의 정보를 수신하고 송신하는 역할을 하는 모듈로서 메시지 기반 통신을 한다. 여기서 사용되는 메시지는 RTSP를 바탕으로 정의한 SCP이다. SCP는 응용 계층에 존재하는 VSS와 스트리밍 서버 응용간의 정보 관리 및 제어를 위해 설계된 프로토콜이다. 세션 관리 모듈에서 이러한 SCP의 메시지 형식을 정의하고 있으며, 이를 처리하기 위한 서버와 클라이언트 인터페이스를 제공해 준다. SCP에서의 메시지 설계는 RTSP를 바탕으로 하였으나, 참여자에 대한 정보와 권한을 프로토콜 내에서 명시하고 있다는 점과, 오직 연결지향형 서비스만을 지원한다는 점이 다르다. SCP는 스트리밍 서버의 정보 공유만이 그 목적이 아니며 제어에 관한 부분도 고려하여야 되기 때문에 참여자에 대한 권한 명시와 보안과 신뢰성을 위해 연결 지향형 서비스를 지원하고 있다.

정보 관리 모듈은 세션 제어 모듈과 동작 구조가 동일하며, 이 모듈을 이용하는 것은 스트리밍 서버의 컴퓨팅 부하를 감소시키고, 클라이언트에 원활한 정보를 제공하였다. SCP 메시지의 메소드 기능 명세는 다음의 표 1과 같다. 그림 13은 SCP 프로토콜을 사용하여 스트리밍 미디어의 정보와 서버의 정보를 등록하고 요청하는 알고리즘을 보여준다. 이 등록 및 요청 처리 알고리즘은 두 가지의 종류로 구분되어 동작된다.

SCP를 통한 정보 교환은 표 1의 Method의 형식을 통해 이루어진다. 크게 정보를 얻어오는 기능(REQUEST)과 정보를 등록, 제거하는 기능(REGISTER, UNREGISTER)으로 나눌 수 있다. REQUEST는 미디어의 정보, 서버의 정보 등을 클

표 1. SCP의 메소드 기능

메소드 이름	메소드 기능
REQUEST_CONTENT_LIST	VSS에 등록된 서버의 컨텐츠 정보를 받아올 때 사용한다.
REGISTER_CONTENT	VSS에 서버의 컨텐츠 정보를 설정할 때 사용한다.
REQUEST_CHANNEL_LIST	현재 서비스되는 채널(세션)의 정보를 받아올 때 사용한다.
REGISTER_CHANNEL	채널의 생성이나 존재하는 채널의 상태를 바꿀 때 사용한다.
REGISTER_SERVER	VSS에 서버의 정보를 설정할 때 사용 한다.
REQUEST_HOSTINFO_LIST	VSS에 등록된 서버의 정보를 받아올 때 사용한다.
REGISTER_ALIAS	서버의 방송 장르를 등록할 때 사용한다.
REQUEST_ALIAS_LIST	등록된 방송 장르를 받아올 때 사용한다.
UNREGISTER_ALIAS	서버가 중지될 때 등록된 방송 장르의 정보를 VSS에서 제거할 때 사용한다.
UNREGISTER_CONTENT	서버가 중지될 때 등록된 컨텐츠의 정보를 VSS에서 제거할 때 사용한다.
UNREGISTER_CHANNEL	서버가 중지될 때 등록된 채널의 정보를 VSS에서 제거할 때 사용한다.
UNREGISTER_HOST	서버가 중지될 때 등록된 서버의 정보를 VSS에서 제거할 때 사용한다.

```
Communication by SCP Method
   If there are user's requirement
        Switch requirement
// same case : REQUEST_CONTENT ,
// REQUEST_CHANNEL_LIST
// REQUEST_CHANNEL, REQUEST_HOSTINFO,
// REQUEST_ALIAS, REQUEST_ALIAS_LIST
// communication with Client between VSS System

line 1    Case REQUEST_CONTENT_LIST:
line 2      send a requirement to the server media list
 manager(VSS);
line 3      send a requirement to the server session manager;
line 4      initialize session (VSS and client);
line 5        IF connection is established and Server information
 is existed
              {
line 6        initialize the VSS manager;
line 7        the server send a media information to the client;
              }
// same case : REGISTER_CHANNEL, REGISTER_ALIAS,
// UNREGISTER_HOST   UNREGISTER_ALIAS,
// UNREGISTER_CONTENT,
// UNREGISTER_CHANNEL
//communication with Streaming Server between VSS System
line 8    Case REGISTER_SERVER :
line 9    send a requirement to the server information
manager;
line 10   send a requirement to the server session manager;
line 11   IF server is already existing   {
line 12    update the server information in session manager;
line 13      update the requirement server streaming media
list;
            }
             ELSE{
line 14  register the server information in session manager;
line 15  register the requirement server streaming media
list;
            }
```

그림 13. SCP 프로토콜을 사용한 정보 교환 알고리즘

라이언트가 요청할 때 이루어지는 것으로 서버는 VSS시스템에서 요청된 정보를 찾아 클라이언트에 보낸다. 이 정보를 가지고 클라이언트는 스트리밍 서버에 스트리밍 요구를 하게 된다. REGISTER와 UNREGISTER는 미디어 정보, 서버의 정보를 스트리밍 서버가 새로 동작되거나 동작된 이후에 정보가 갱신됐을 때 VSS와 스트리밍 서버와의 정보 교환을 하는 기능으로 VSS시스템의 정보를 갱신하거나 새로운 정보를 등록하는 것을 담당하고 있다.

세부적인 동작을 보면 그림 13과 같이 사용자나 서버 시스템의 요청이 있는 경우 요구사항이 무엇인지 비교하여(line 1, line 8), 사용자의 REQUEST이면 VSS서버의 미디어 리스트 관리 모듈에게 사용자 요청을 보낸다(line 2). 이 요청은 VSS서버의 세션 관리 모듈에 전해지고, 세션 모듈은 VSS와 클라이언트간의 세션을 초기화 한다(line 3, line 4). 세션이 성립되면 VSS 관리 모듈을 초기화하고(line 6), 요청된 정보를 클라이언트에 보낸다(line 7). 요구 사항이 REGISTER이면(line 8), 서버의 정보 관리 모듈과 세션 모듈에 정보를 보낸다(line 9, line 10). 만약, 이전에 서버가 존재하면(line 11), 새로 요청 정보를 통하여 기존의 서버정보와 미디어 리스트를 갱신한다(line 12, line 13). 서버의 정보가 존재하지 않으면, 서버의 정보를 세션 관리 모듈에 등록하고(line 14), 서버의 미디어 정보 역시 미디어 리스트에 등록한다(line 15).

Ⅵ. 성능평가

1. 성능 평가 시스템 모델 및 환경

본 논문에서 제안된 시스템 구조를 이용하여 구현한 MPEG-2 AAC 오디오 스트리밍 시스템은 오디오의 관리와 전송 프로토콜, 전송 에러의 복구와 관련한 QoS기술을 제공하며 서비스와 시스템 관리의 편리성을 위하여 정보관리 기능을 제공한다. 이 시스템을 이용하여 주문형 오디오 서비스와 같은 스트리밍 응용 프로그램을 만들 수 있으며 이렇게 제작한 스트리밍 응용 프로그램은 쉽게 스트리밍 서비스할 수 있다.

그림 14는 본 논문에서 제안한 스트리밍 시스템의 구조를 이용하여 구축한 서버와 클라이언트, 정보관리 서버로 구성된 시스템의 구현 모델을 보여준다. 표 2는 실험을 위한 동작 환경이다.

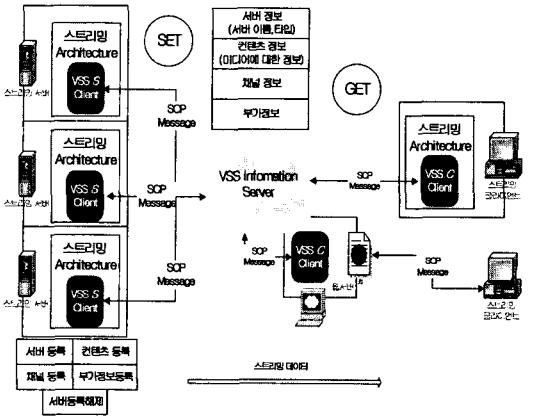

그림 14. MPEG-2 AAC 오디오 스트리밍 시스템을 이용한 시스템 구현 모델

표 2. 서버, 클라이언트, 정보관리 프로그램의 실험 동작 환경

구분 / 항목	서 버	클라이언트	정보관리
운영체제	Windows 계열		
하드웨어	CPU 833 MHz RAM 512MB	CPU 400 MHz(Notebook) RAM 64MB	
전송 프로토콜	RTP(Real-time Transport Protocol) (UDP / TCP)		
네트워크	TCP/IP over 10/100 Mbps LAN		
개발언어	C++ Language		

2. 성능 평가 결과 및 분석

성능평가 결과 오디오 스트리밍의 네트워크 전송 및 재생의 기능과 QoS 지원 기능이 바르게 동작하는지 여부를 확인할 수 있었다. 하지만, FEC와 재전송을 고려한 QoS 모듈의 동작과 관련한 네트워크 오버헤드에 대한 분석은 미비하였다. 그림 15는 MP3파일 포맷을 이용한 스트리밍과 MPEG-2 AAC 파일 포맷을 사용한 스트리밍 서비스의 네트워크 자원 점유율 그래프로서 MPEG-2 AAC의 오디오 데이터 스트리밍이 상대적으로 약 1.5%의 적은 네트워크 자원을 사용하는 것을 알 수 있다. 또한, 파일포맷의 특성으로 인하여 네트워크 자원을 적게 사용하는 것은 재전송으로 인한 네트워크 부하 증가의 문제를 해결하여 주고 있다. 이는 기존의 시스템과는 비슷한 네트워크 자원을 점유하면서 서비스의 질적 향상과 안정성을 높이는 효과를 보였다. 산술적으로 패킷 손실이 1초에 한번 일어났을 경우 재전송으로 인한 패킷의 네트워크 부하는 전체 네트워크 자원의 5%를 넘지 않으며 이 점유율과 전체 점유율을 합해도 MP3오디오 스트리밍의 네트워크 점유율을 넘지 않는 것을 볼 수 있다.

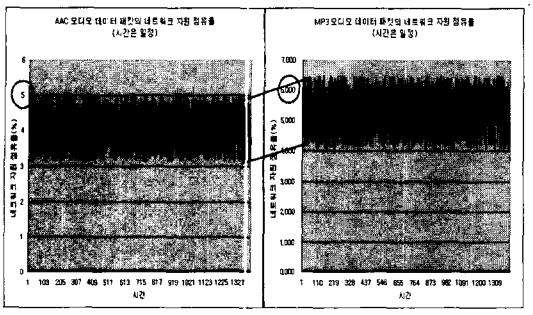

그림 15. AAC오디오와 MP3 오디오 데이터의 네트워크 점유율

본 논문에서 제안한 시스템은 기본적으로 재전송 기반의 QoS 전송 모듈을 사용하고 있는데, 재전송은 패킷의 새로운 전송으로 인한 네트워크 자원 점유의 오버헤드를 피할 수 없다. 하지만 재전송을 사용하지 않았을 때 기존의 MP3 스트리밍 시스템과 AAC 스트리밍 시스템의 네트워크 자원 점유율을 비교하면 MP3 시스템이 비해 AAC 스트리밍 시스템이 적은 것을 알 수 있다. 따라서 제안된 시스템은 AAC의 사용으로 재전송 기법의 네트워크 오버헤드를 상쇄하면서, 나아가 MP3보다 더 나은 음질의 AAC 오디오를 스트리밍 할 수 있는 스트리밍 시스템이다.

그림 16은 오디오 패킷의 손실에 따른 데이터 전송의 네트워크 점유율 변화를 나타내는 것으로 패킷 손실의 양(손실 패킷의 개수)에 따라 네트워크 전송을 위한 데이터양과 네트워크 전송 점유율이 변화하는 결과를 보여준다. 평균적인 데이터 전송은 500Kbit-per-second로 가정하고, 네트워크의 변화가 없을 때의 수치이다.

재전송으로 인한 네트워크 점유율을 살펴보면 패킷의 손실에 따라서 데이터 전송 폭이 커지는 것을 볼 수 있는데, 이는 재전송 패킷이 일반적인 패킷을 보내는 네트워크 자원과 상관관계 없이 전송되기 때문이다. 네트워크 오버헤드는 사용자에게 신뢰성을 주지 못하고, 스트리밍 서비스의 응답시간을 느려지게 만드는 요인이 된다. 본 논문에서 제안한 시스템은 이처럼 충분한 네트워크 속도와 자원이 지원되지 않아 재전송 오버헤드가 스트리밍 서비스 상의 큰 문제로 작용하게 되면, 시스템은 FEC 패킷 방법을 사용하여 스트리밍 데이터를 전송하는 방법을 쓰게 된다. 이 FEC 방법을 쓰면 음질이 저하되는 단점을 가지고 있지만 스트리밍 서비스의 연속성을 유지하고, 네트워크의 오버헤드를 줄이고, 일정하게 유지할 수 있으며 응답시간을 줄일 수 있다는 장점을 가질 수 있어 유연성을 제공해 준다.

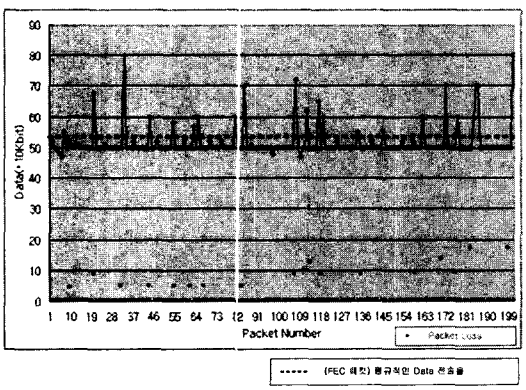

그림 16. AAC 오디오의 패킷 손실에 따른 전송(재전송) 네트워크 점유율 변화

3. 토 의

오디오 스트리밍 시스템의 설계 및 구현, 성능 평가의 결과로 MPEG-2 AAC 오디오가 MP3오디오 포맷보다 네트워크 점유율 부분에서 높은 성능을 보여 주었다. 따라서, 제안된 시스템은 낮은 전송속도를 지닌 네트워크 환경에서 사용하기에 적합하다. 또한, 패킷 손실이나 에러의 보정에 대한 고려가 되어있어 유선 네트워크 환경보다 패킷 손실률이 높은 무선 네트워크 환경의 경우에도 쓰일 수 있을 것으로 판단된다. 하지만 이 경우에도 다양한 환경 변수(네트워크 대역폭, 에러율 등)등을 통한 QoS 정책에 변화를 줄 수 있기 때문에 유연한 구조를 그대로 상속받게 된다. 복수개의 서버가 동작되는 경우, 정보 관리 매커니즘을 도입하여 기존의 시스템이 정적인 정보를 제공하고 정보 갱신이 느린 점을 보완하였고, 서버 정보와 미디어 정보의 원활한 처리, 동적 정보제공 및 시스템 관리를 편리하게 만들었다. 하지만, 이런 장점에도 불구하고 본 논문에서 제안된 시스템은 네트워크 자원의 다양한 환경적 요소가 고려되어 있지 않다는 제약점을 가지고 있다.

Ⅶ. 결 론

제한적인 네트워크 자원을 사용할 수밖에 없는 상황에서 오디오 데이터를 네트워크로 전송하여 실시간으로 재생하는 오디로 스트리밍을 위해서는 스트리밍 데이터의 전송과 제어, QoS를 위한 시스템 구조가 요구된다. 또한, 복수개의 서버의 원활한 동적 정보 관리를 위하여 정보를 관리, 제공하고 스트

리밍 서버의 부하를 줄일 수 있는 정보관리 기술이 요구된다. 이러한 관점에서 제안된 시스템은 여러 스트리밍 기능을 제공하는 시스템 구조를 설계하고, 각 모듈을 객체지향적인 모델로 설계 구현하였다.

인터넷을 통한 오디오 스트리밍 서비스는 보다 낮은 용량의 데이터를 보내야만 더 나은 서비스를 할 수 있게 되는데, 이점에서 MPEG-2 AAC 오디오 포맷은 현실의 네트워크에 가장 적합한 멀티미디어 포맷이라 할 수 있겠다. 제안한 오디오 스트리밍 시스템은 네트워크의 가변적인 전송률 및 속도에 따른 패킷손실로 인한 서비스의 불안정을 해소하기 위하여 재전송과 FEC기법 등 QoS기법을 포함시켰으며, 스트리밍 정보의 동적인 제공을 위하여 기존 스트리밍 제어 프로토콜과 유사한 구조의 정보관리 프로토콜을 제안하고 구현하였다. 그리고, 본 논문에서 다루는 스트리밍 시스템은 네트워크 점유율 검증을 통하여 기존의 MP3를 사용한 오디오 스트리밍 시스템과의 비교에서 비교우위를 보였으며, 패킷 손실로 인한 서비스 품질의 저하를 방지하고, 안정화에 기여를 하고 있다. 또한, 정보관리 기법은 클라이언트에게 동적인 스트리밍 정보를 제공하여 서비스의 질을 높이는 결과를 가지고 왔다. 한편, 제안된 오디오 스트리밍 시스템은 오디오를 기반으로 하여 스트리밍 시스템을 설계하였지만, 비디오와 같은 다른 멀티미디어 포맷을 지원할 수 있는 유연한 구조를 가지고 있다.

그러나, 제안된 MPEG-2 AAC 오디오 스트리밍 시스템은 적용된 QoS기법에 있어 재전송에 의한 네트워크 오버헤드 및 FEC 방법에 의한 서버 시스템의 오버헤드가 발생하고 있다. 이점을 해결하기 위해서는 패킷의 무결성을 보장하는 기술과 네트워크 부하를 최소화시키기 위해 데이터의 압축률을 높이는 연구가 필요하다. 또한 오디오 패킷 전송의 성능을 발전시키기 위해서는 네트워크의 대역폭을 분석하고 모니터링 하여 알맞은 패킷과 적합한 QoS 기법을 적용해야 할 필요가 있으며, 정보관리 기법을 더욱 발전시켜 더욱 정확한 정보를 제공하는 방안에 대한 구체적인 연구가 요구된다. 그리고, 다양한 미디어에 대한 확장성에 대한 성능평가가 더 이루어져야 되고, 기존 시스템이 처리할 수 있는 처리량과의 비교 분석도 더 이루어져야 할 것이다.

참 고 문 헌

[1] Taejin Lee, Jose Soler Lucas, Jinwoo Hong, "Streaming of AAC data over Internet by using RTP/RTCP", in *17th ICA*, ROME, 2001 Sep.

[2] Microsoft. "Windows Media Technologies Technical Overview". Technical White Paper. http://www.microsoft.com/ntserver/techresources/streaming/default.asp

[3] J. Rosenberg and H. Schulzrinne. "An RTP Payload Format for Generic Forward Error Correction". RFC2733, Internet Engeering Task Force, December 1999

[4] W. Stevens, "UNIX Network Programming, Volume 1, Second Edtion: Networking APIs: Sockets and XTI," Prentice Hall, 1998, ISBN 0-13-490012-X.

[5] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, IETF RFC 1889, January 1996, ftp://ftp.isi.edu/in-notes/rfc1889.txt.

[6] H. Schulzrinne, A. Rao, R. Lanphier, Real-Time Streaming Protocol (RTSP), IETF RFC 2326, April 1998, ftp://ftp.isi.edu/in-notes/rfc2326.txt.

[7] K. Mayer-Patel and L.A. Rowe, "Design and Performance of the Berkeley Continuous Media Toolkit", *Multimedia Computing and Networking* 1997, pp 194-206, 1997.

[8] Real-Networks, Real-Systems, http://www.real.com/

[9] C. Perkins, O. Hodson and V. Hardman, "A Survey of Packet Loss Recovery Techniques for Streaming Audio," *IEEE Network Magazine*, September/October 1998, pp.40-47

[10] D. Meares. K. Watanabe, E. Scheirer, "Report on the MPEG-2 AAC Stereo Verification Tests," ISO/IEC JTCI/SC29/WG11 N2006, 1998, Feb.

[11] K. Jonas, M. Kretschmer, and J. Mödeker, "Get a KISS - Communication Infrastructure for Streaming Services in a Heterogeneous Environment", *In Proc. of ACM Multimedia '98*, Bristol, UK, pp. 401-410, 1998.

[13] Shanwei Cen, Calton Pu, Jonathan Walpole, "Flow and Congestion Control for Internet Media Streaming Applications", *Tech Report in Oregon Institute of Science and Technology*, Mar. 1997

[14] R. Field, UC. Irvine, "HyperText Transfer

Protocol - HTTP 1.1", IETF RFC 2068, Jan. 1997

[15] Nullfost "Shoutcast System Overview," Technical web page.

이 승 재(Seung-Jae Lee) 준회원

2002년 2월 : 경희대학교 컴퓨터공학과(공학사)

2002년 3월~현재 : 경희대학교 컴퓨터공학과 석사과정

<주관심 분야> 멀티미디어, 분산 시스템, 미들웨어

이 승 룡(Sung-Young Lee) 정회원

1978년 2월 : 고려대학교 재료공학과(공학사)

1986년 12월 : Illinois Institute of Technology 전산학과 석사

1991년 12월 : Illinois Institute of Technology 전산학과 박사

1991년 9월~1993년 8월 : Governors State University 조교수

1993년 9월~1996년 9월 : 경희대학교 전자계산공학과 조교수

1996년 10월~2001년 9월 : 경희대학교 전자정보학부 컴퓨터공학전공 부교수

2001년 10월~현재 : 경희대학교 전자정보학부 컴퓨터공학전공 교수

<주관심 분야> 실시간 컴퓨팅, 실시간 미들웨어, 멀티미디어 시스템, 시스템 보안, 분산 시스템