스트리밍 프레임워크에서 미디어 관리자의 설계 및 구현

(Design and Implementation of Media Manager in Multimedia Streaming Framework)

이 재 욱 † 이 승 룡 †† 홍 인 기 †††

(Jaewook Lee) (Sungyoung Lee) (Een-Kee Hong)

요 약 본 논문에서는 멀티미디어 스트리밍 프레임워크에서 미디어 관리자의 설계와 구현에 대한 경험을 기술한다. 미디어 관리자는 스트리밍 프레임워크 내에서 미디어 스트림이 어떠한 타입의 소스로부터 얻어지며, 그것이 어떠한 종류의 스트림인가를 판별하고, 획득된 미디어를 가장 적절하게 처리할 수 있는 코덱을 선택하며, 어떠한 미디어 디바이스를 통해 재생되어야 효과적인지를 식별하고 관리하기 위해서 필요하다. 제안된 미디어 관리자는 크게 미디어 소스와 싱크 모듈로 구성되어 있는데, 미디어 소스 모듈은 미디어를 추상화시킴으로써 여러 소스로부터 입력되는 성격이 다른 미디어들을 어떤 소스에서 전달된 미디어인지 상관하지 않고 효과적이고 일관된 방법으로 처리할 수 있다. 미디어 싱크 모듈은 클라이언트 측에서 얻어온 미디어 데이타를 적절한 미디어 디바이스에 분배해주는 역할과 전달된 미디어를 다양한 미디어 표현장치를 통해 재생시키는 역할을 수행한다. 제안된 미디어 관리자는 멀티미디어 데이타베이스와 연동기능을 지원함으로써 높은 부가가치 서비스 제공을 가능케 하였고, RTP/RTSP 소스필터나 Winamp 게이트웨이 기능도 지원함으로써 융통성을 제공한다. 더욱이, 향후 새로운 형태의 미디어 소스가 출현하더라도 이를 용이하게 스트리밍 프레임워크에 추가시켜 서비스할 수 있는 유연성과 확장성을 지원한다.

Abstract In this paper, we introduce our experience for designing and implementing a media manager in the Integrated Streaming Service Architecture (ISSA) developed by the authors. The media manager is regarded as a necessary module in the ISSA framework for the following reasons. It realizes that from which locations of the media source devices, the media streams are coming. Once it knows where the origin is, the media manager should recognizes what types of stream are. After that, it performs how to chose an appropriate CODEC to handle the recognized input streams efficiently, and what type of media playback device should be selected. In order to do such a job efficiently, the proposed media manager consists of two modules; source module and sink module. The major role of a media source module is to make an abstraction for the media streams that are coming from various types of media device. This, in consequence, enables a media manager to consistently handle the media streams without considering wherever they come from. On the other hand, the media sink module distributes the input streams to an appropriate media device to playback. One of the remarkable virtues of the proposed media manager is an ability to supporting high value-added database services since it provides an interface between the ISSA and real-time multimedia database. Also, it provides the RTP/RTSP source filter and Winamp gateway modules which allow the flexibility to the system. Moreover, the media manager can adopt any types of new media which in fact will provide scalability to the ISSA.

† 비 회 원 : 경희대학교 전자계산공학과
jaeki@oslab.kyunghee.ac.kr
†† 종신회원 : 경희대학교 전자계산공학과 교수
sylee@oslab.kyunghee.ac.kr
††† 비 회 원 : 경희대학교 전자정보학부 교수
ekhong@khu.ac.kr
논문접수 : 2000년 3월 3일
심사완료 : 2001년 5월 21일

1. 서 론

최근 디지털 미디어 관련 기술의 동향은 인터넷을 중심으로 발전하고 있지만, 디지털 TV와 함께 IMT 2000과 같은 무선 통신 환경에서도 작동하는 디지털 미디어에 대한 연구 개발 또한 활발히 진행중이다. 이러한 상

황에서 디지털 미디어를 인터넷, 초고속 통신망, 방송망, 무선 통신과 같은 다양한 환경에 쉽게 적응시킬 수 있는 통합 스트리밍 프레임워크 기술이 요구되고 있다. 스트리밍이란 통신망에서 음성, 영상과 같은 멀티미디어 데이타를 전송할 때 실시간으로 재생이 이루어져 사용자에게 마치 생생하게 대화하는 듯한 환경을 제공해 줄 수 있는 기술을 말한다. 그러나, 지금까지의 스트리밍 프레임워크에 관한 대부분의 연구는 다른 시스템과의 연동을 고려하지 않고 독자적인 환경에서 동작하도록 설계되어 전체 프레임워크의 유연성이 부족하며 확장이 용이하지 않다[1],[2],[3]. 또한 다양한 미디어를 지원하기 위한 오디오/비디오 Codec의 지원기능이 미약하며, 이 기종 운영체제, 네트워크 환경을 지원할 수 있는 투명성도 부족하다.

이 같은 제한점 개선을 위하여 다른 스트리밍 시스템과의 연동이 용이할 뿐만 아니라, 다양한 오디오/비디오 미디어 형식을 지원하고 이 기종 운영체제와 네트워크 환경에서 동작할 수 있는 적응력을 지닌 통합 멀티미디어 스트리밍 구조인 ISSA(Integrated Streaming Service Architecture)를 저자는 제안하였다[4]. ISSA는 RTP(Real-Time Transport Protocol)[5], RTSP(Real-Time Streaming Protocol)[6]와 같은 표준 실시간 전송 프로토콜을 사용함으로써 사용자에게 범용성을 제공하며, 멀티미디어 실시간 데이타 베이스인 BeeHive[7]와의 연동 기능도 제공하고 있다. 또한, ISSA는 유니캐스팅/멀티캐스팅 환경의 VOD(Video on Demand) 시스템과 실시간 방송시스템(라이브캐스팅)과 같은 멀티미디어 스트리밍 서비스 응용을 개발하기 위한 라이브러리를 제공한다.

본 논문에서는 ISSA의 주요 모듈인 미디어 관리자의 설계와 구현에 대한 개발 경험을 소개한다. 제안된 미디어 관리자는 크게 미디어 소스와 미디어 싱크 모듈로 구성되어 있다. 미디어 소스 모듈은 미디어를 추상화시킴으로써 성격이 다른 여러 가지의 소스로부터 입력되는 미디어를 어떤 소스에서 전달된 것인지 상관하지 않고 효과적이고 일관된 방법으로 미디어를 처리할 수 있다. 미디어 싱크 모듈은 클라이언트 측에서 얻어온 데이타를 적절한 미디어 디바이스에 분배해주는 역할과 전달된 미디어를 다양한 미디어 표현장치를 통해 재생시키는 역할을 수행한다. 미디어 싱크 모듈에 전달된 미디어는 다양한 미디어 표현방법을 이용하여 재생되는데, 이러한 표현방법은 Push/Pull을 동시에 지원하는 클라이언트의 버퍼 관리자를 이용, COM(Component Object Model)기반의 DirectShow[8]와 Winamp Gateway를 통해서 다양한 미디어 데이타를 재생할 수 있는 기능을 제공한다.

기존의 미디어 관리자들은 Codec 구현 시 독립적인 전송 프로토콜의 송수신 모듈에 DirectShow 또는 ActiveMovie 기술의 트랜스폼 필터를 연결한 단순한 구조가 주류이지만 제안된 미디어 관리자는 파일, 데이타베이스 또는 라이브 비디오 카메라와 같이 여러 형태의 소스로부터 얻어낸 미디어를 일관성있게 추상화시킬 뿐만 아니라, ISSA 프레임워크의 RTP/RTSP 프로토콜을 처리할 수 있는 DirectShow 기술의 소스 필터와 Winamp에서 RTP 프로토콜을 지원하는 Plug-in인 Winamp Gateway를 개발하여 확장성과 사용자의 편의성을 제공해준다. 또한 라이브 비디오 카메라와 마이크를 이용하는 실시간 화상회의 미디어 스트림, 파일 시스템 내에 있는 미디어 파일, 멀티미디어 데이타베이스의 관리를 받는 미디어까지 사용할 수 있을 뿐만 아니라, 향후 새로운 형태의 미디어 소스가 출현하더라도 이를 용이하게 스트리밍 프레임워크에 수용하여 서비스할 수 있는 유연성과 확장성을 가지고 있다.

본 논문의 구성은 다음과 같다. 제2장에서는 본 논문이 제안하는 미디어 관리자와 유사한 관련 연구를 소개하고, 제3장에서는 본 논문에서 제안하는 미디어 관리자를 사용하는 분산 스트리밍 프레임워크에 대해서 간략히 설명한다. 제4장에서는 미디어 관리자의 구성과 기능 그리고 각 세부모듈의 설계에 대하여 기술하고, 제5장에서는 미디어 관리자의 구현 사례를 보여준 뒤, 제6장에서 결론을 내린다.

2. 관련 연구

본 장에서는 스트리밍 시스템에서 미디어 관리자에 대한 기존의 연구에 대해 살펴보겠다. JMF(Java Media Framework)[9]는 SUN 사에서 출시한 API로 자바에서 다양한 음악 파일과 동영상 파일을 재생 할 수 있게 해주며, 컴포넌트화된 구조를 가지고 있다. JMF에서의 미디어 관리는 네트워크나 로컬에 위치한 파일을 소스로 채택할 수 있으며, 데이타의 흐름을 서버 쪽에서 관리하고 클라이언트 쪽으로 데이타를 밀어 넣는 프로토콜인 RTP 방식과, 데이타의 흐름을 클라이언트 쪽에서 관리하며 서버 쪽에 미디어를 요청해서 가져오는 방식인 HTTP나 FILE 프로토콜의 두 가지 방식을 모두 지원한다. 그리고 JMF는 다양한 포맷의 미디어 파일을 플랫폼에 독립적으로 실행시킬 수 있는 장점을 가지고 있다. 그러나, 미디어 소스 채택 면에서는 멀티미디어 데이타베이스와 연동 기능을 제공하지 못한다는 점, 전

송 부문에서는 UDP 기반의 RTP와 TCP/IP등 다양한 네트워크 환경에 대응하지 못하는 점이 본 논문에서 제안한 미디어 관리자와 다르다.

[10]에서는 고속 전후진 기능을 제공하는 웹기반 VOD 시스템을 개발하였는데, 이 시스템은 웹을 이용한 사용자 인터페이스, 실시간 데이타 전송을 위한 서버, 미디어를 재생하기 위한 클라이언트로 구성되어 있다. 서버는 클라이언트가 요구하는 연속적인 데이타를 네트워크를 통해 제한된 시간 내에 전송하기 위하여 실시간 스케쥴러를 사용하였고, 클라이언트는 스트림 방식으로 MPEG-1 동영상 데이타를 전송 받도록 설계되었으며, ActiveMovie를 이용하여 재생 할 수 있도록 구현되었다. 이 시스템은 자체적으로 개발한 알고리즘을 이용하여 동영상을 시청 중에 고속 전 후진이 가능한 서비스를 제공할 수 있는 특징을 가지고 있다. 그러나, 이 시스템은 미디어의 관리측면에서 소스 채택 시 로컬 파일 시스템에서 읽어오는 방식만 있다는 점과, ActiveMovie를 지원하지만 DirectShow의 기준을 따르는 소스 필터를 제작하지 않아 다양한 미디어 형식을 지원하지 못하여 확장성이 떨어진다는 점이 본 논문에서 제시하는 미디어 관리자와 다르다.

3. 스트리밍 프레임워크의 기능과 구조

이 장에서는 ISSA 프레임워크에 대하여 간단히 소개한다. ISSA 프레임워크의 모델은 [그림 1]과 같이 크게 상위 계층의 스트리밍 응용, ISSA와 스트리밍 응용 사이의 인터페이스인 MOA(Media Object Architecture)[11]. 데이타베이스 커넥터 모듈, 게이트웨이 모듈(Gateway Module), 그리고 ISSA로 구성되어있다. MOA는 다시 응용 인터페이스(Application Interface)와 응용 Wrapper로 구성되어 진다. 응용 인터페이스는 AMS(Agent for Multimedia Serveice)와 스트리밍 서버(Streaming Server)로 나누어지는데, AMS의 역할은 각 스트리밍 서버에 분산되어 있는 컨텐츠의 정보나 세션의 정보를 통합, 분배하는데 있다. 스트리밍 서버의 역할은 모듈화되어 있는 RTTP(Real-Time Transaction Protocol) 서버나 RTSP 서버 모듈들의 통합 기능과 AMS에게 스트리밍 서버 자신이 가지고 있는 컨텐츠나 이미 존재하는 세션을 알려주는 역할을 한다. 그리고, 응용 Wrapper는 DirectShow 소스 필터(Source Filter)와 Winamp 플러그 인(Plug-in)으로 구성된다. Direct Show 소스필터는 Microsoft 사의 Windows Media Player와 연동 시에 사용되며, Winamp의 플러그인은 MP3 방송 서비스를 위해 사용된다. 또한, 데이타베이스 커넥터는 실시간 멀티미디어 데이타베이스인 BeeHive와 연동시켜주는 BeeHive 커넥터, 게이트웨이 모듈은 CORBA연동을 위한 CORBA 게이트웨이로 구성된다.

그리고, 프레임워크의 핵심을 이루는 ISSA의 기본구조는 세션, 전송, 미디어 및 자원 관리자로 구성되어 있다. 세션 관리자는 RTSP 기반의 멀티미디어 스트리밍 서비스의 세션 제어를 담당하며, 유니캐스팅과 멀티캐스팅을 지원할 수 있는 구조로 되어 있다. 또한, 데이타베이스의 트랜잭션 요청을위해 RTTP, 컨텐츠 정보의 분배와 공유를 담당하는 SCP(Service Control Protocol)를 지원한다. 전송 관리자는 TCP 및 UDP, 그리고 RTP/UDP 프로토콜을 이용하여 멀티미디어의 전송을 담당하며, RTCP 프로토콜을 이용하여 네트워크의 상태를 모니터링 한다. 미디어 관리자는 미디어의 인코딩과 디코딩을 담당하며, 현재 MPEG-1, MPEG-2, ASF, MP3를 지원한다. 자원관리자는 스트리밍 시스템에서 QoS의 명세화, 매핑, 모니터링, 제어 기능을 제공하며, 메모리 버퍼 관리, 쓰레드 스케줄링 등의 기능을 수행한다[12],[13].

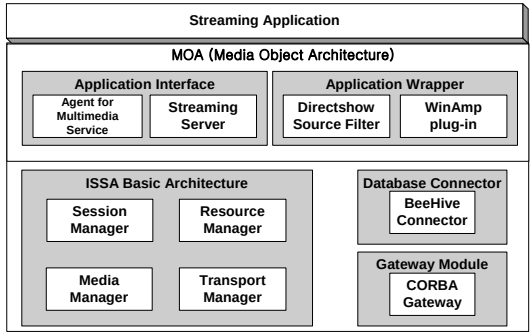

그림 1 ISSA 스트리밍 프레임워크의 구조

4. 미디어 관리자 설계

4.1 미디어 관리자의 구조

ISSA에서 미디어 관리자는 파일, 멀티미디어 데이타베이스, 그리고 라이브 동영상을 위한 비디오나 마이크와 같은 디바이스로부터 얻어진 미디어를 추상화시킨 다음 이를 UDP기반의 RTP/RTSP를 사용하는 전송 관리자에게 넘겨주고, 그리고 전송 관리자를 통해서 받아온 미디어를 가장 효율적인 디바이스를 통해서 재생하는 기능을 제공해 준다.

전술한 바와 같이 미디어 관리자는 크게 미디어 소스와 미디어 싱크의 두 가지 모듈로 구성되어 있다[그림

2]. 미디어 소스 모듈의 가장 중요한 기능은 입력된 미디어를 추상화시키는 것으로써, 멀티미디어 데이타베이스인 BeeHive나, 실시간 동영상을 위한 디바이스와 같이 성격이 다른 여러 소스로부터 얻어지는 미디어를 일관된 방법으로 처리할 수 있는 장점이 있다. 미디어 싱크 모듈은 클라이언트 측에서 받아온 미디어 데이타를 가장 적당한 디코딩 방법을 선택할 수 있는 분배기 역할을 한다. 그리고, 클라이언트 측에서 전송 관리자로부터 수신된 미디어 데이타를 넘겨받아 적절한 오디오/비디오 장치로 재생하는 역할 또한 가지고 있다.

현재 지원 가능한 오디오/비디오의 종류는 MPEG-1, MPEG-2, MP3, WAVE, AU, ASF 등의 다양한 미디어들이 있으며, 제안된 미디어 관리자는 DirectShow 기술을 사용하여 대부분의 미디어 디코딩 부분을 처리하지만, DirectShow에서 제공하지 못하는 MPEG-2와 같은 형태의 미디어의 경우에는 그 형태의 포맷을 처리할 수 있는 다른 외부 모듈을 이용하고, 서비스 중 그러한 포맷의 스트림을 요구받으면 미디어 싱크 모듈은 외부 모듈과 요구된 스트림을 연결시켜주는 역할을 한다.

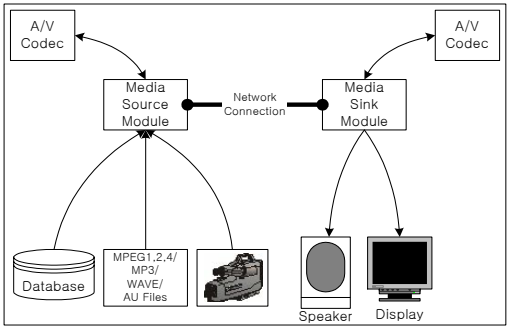

그림 2 미디어 관리자의 구조

제안된 미디어 관리자는 라이브 비디오 카메라와 마이크를 사용하는 실시간 화상회의 미디어 스트림, 파일 시스템의 미디어 파일, 멀티미디어 데이타베이스의 관리를 받는 미디어까지 처리할 수 있을 뿐만 아니라, 향후 새로운 형태의 미디어 타입이 출현하더라도 이를 쉽게 수용할 수 있는 유연성과 확장성을 가지게 된다.

미디어 관리자의 클래스 다이어그램은 [그림 3]에서 보여준다. 미디어 관리자를 각각의 큰 모듈로 나누어 보면, 미디어 소스 모듈, 미디어 정보 모듈, 미디어 싱크 모듈, 버퍼 관리자로 나눌 수 있다. 미디어 소스 모듈은 파일, 데이타베이스, 마이크와 같은 다양한 미디어 데이

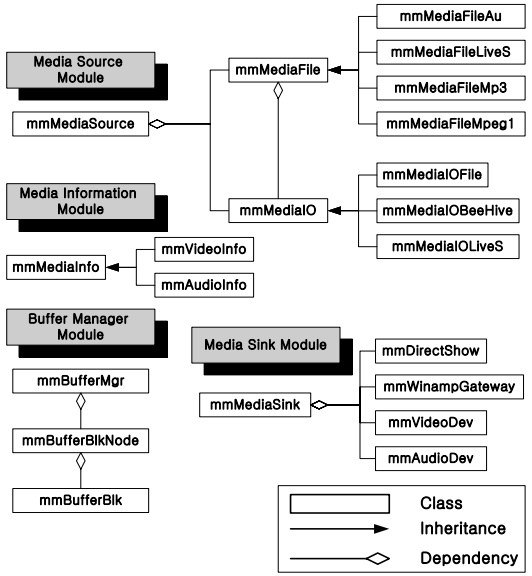

그림 3 미디어 관리자의 클래스 다이어 그램

타를 저장 형태에 따라 적절한 방법으로 전송할 수 있도록 만드는 역할을 한다. 미디어 정보 모듈은 인코딩 형식, 저장 방식 등의 미디어 관리자에서 필요한 미디어의 정보들을 관리한다. 버퍼관리자는 서버로부터 전송되어진 미디어 스트림을 클라이언트에서 효과적으로 버퍼링하기 위한 모듈이다. 미디어 싱크부분은 클라이언트 부분에서 미디어의 재생을 위한 모듈로써 미디어의 인코딩 형식에 따라서 적절한 재생 소프트웨어를 연동하도록 하는 기능을 한다.

4.2 미디어 관리자 각 모듈의 설계

1) 미디어 소스 모듈

미디어 데이타의 저장 형태에 따라 알맞은 전송 방법을 선택하는 미디어 소스 모듈은 미디어 데이타의 다양한 형식과 저장 형태를 지원하기 위한 추상화 방법으로 mmMediaIO와 mmMediaFile 클래스를 이용한다. 미디어 소스 모듈의 핵심 클래스인 mmMediaSource는 mmMediaIO와 mmMediaFile 클래스를 이용하게 된다. 저장 형태를 추상화하는 mmMediaIO 클래스는 미디어의 저장 형식에 알맞은 처리를 위한 추상화 클래스로 mmMediaIOFile, mmMediaIOBeeHive, mmMediaIOLiveS 클래스를 자식 클래스로 가지고 있다. mmMediaIOFile은 파일로 저장되어진 미디어를, mmMediaIOBeeHive는 BeeHive 데이타베이스에 저장되어진 미디어를, mmMediaIOLiveS는 마이크로 입력되어지는 음성 데이

타를 지원한다. mmMediaFile 클래스는 다양한 미디어들이 각각 알맞은 형태로 전송할 수 있도록 패킷 구성 정보를 추출하고, 미디어 데이타의 인코딩 형식을 판별한다. 이 클래스는 자식 클래스로써 미디어의 인코딩 종류에 따라 mmMediaFileAu, mmMediaFileWave, mmMediaMpeg, mmMediaFileMp3, mmMediaFileLiveS 클래스들을 가지고 있다.

미디어 소스 모듈의 클래스들에 관한 자세한 설명은 다음과 같다.

- mmMediaIO 클래스

이 클래스는 다양한 형태로 저장되어 있는 미디어의 입출력을 통합하여 관리할 수 있도록 하는 추상화 클래스이다. 이 클래스의 파생클래스로는 mmMediaIOFile와 mmMediaIOBeeHive, mmMediaIOLiveS 클래스가 있다. 이들은 각각 파일로 저장 되어 있는 미디어, 멀티미디어 데이타베이스인 BeeHive의 관리를 받는 미디어, 마이크로 입력되어지는 음성 데이타의 입출력을 담당한다. mmMediaIO는 다양한 미디어 소스의 입출력을 하나의 일관된 인터페이스로 관리하는 역할을 한다. 이러한 구조를 사용하면 향후에 추가되는 새로운 형태의 미디어 소스에 대한 입출력 클래스를 수용하는데 있어 mmMediaIO 클래스의 파생 클래스를 추가함으로써 스트리밍 서비스의 범위의 확장 시 쉽게 대응할 수 있으며 mmMediaIO 인터페이스를 이용하여 다양한 형태의 미디어 입출력 방식을 컨트롤 할 수 있다. 예를 들어 실시간 동영상의 처리기능을 수용할 경우 mmMediaIORTAV라는 클래스 이름의 객체를 mmMediaIO 클래스로부터 상속받아 실기간 동영상에 관한 세부 항목을 작성한 후 추가하고, mmMediaIO 인터페이스를 이용하여 제어하면 된다. mmMediaIO의 기본 인터페이스는 [표 1]과 같다.

표 1 mmMediaIO 클래스의 인터페이스

함 수	설 명
Open()	미디어 소스를 오픈한다.
Read()	미디어 소스로부터 미디어 스트림을 읽어온다.
SetPointer()	요구되어지는 미디어의 위치로 이동한다.
GetTotalLength()	미디어의 총 길이를 얻어온다.
Close()	미디어를 닫는다.

- mmMediaIOFile 클래스

mmMediaIOFile 클래스는 미디어 스트림의 입출력을 담당하는 mmMediaIO 클래스를 상속받은 클래스로 로컬에 파일로 저장되어 있는 미디어의 입출력을 담당한다. 이 클래스는 mmMediaIO 클래스와 동일한 인터페이스를 가지고 있으며 mmMediaIO 클래스를 통하여 제어한다.

- mmMediaIOBeeHive 클래스

이 클래스는 mmMediaIOFile과 함께 mmMediaIO 클래스를 상속받는 클래스의 하나로 멀티미디어 데이타베이스인 BeeHive의 관리를 받는 미디어 스트림의 입출력을 담당한다. 이 클래스는 ISSA 프레임워크와 BeeHive와의 연결을 위해 제작되어진 BeeHive Connector의 인터페이스를 이용한다. BeeHive Connector는 BeeHive와의 연결을 위한 인터페이스를 가지고 있고, mmMediaIOBeeHive 클래스는 BeeHive Connector의 인터페이스를 이용하여 멀티미디어 데이타베이스인 BeeHive의 관리를 받는 미디어 스트림에 접근하여 미디어 스트림을 얻어온다. 이 클래스도 mmMediaIOFile과 마찬가지고 mmMediaIO 클래스의 인터페이스와 동일한 인터페이스를 가지고 mmMediaIO 클래스를 통하여 제어한다.

- mmMediaIOLiveS 클래스

이 클래스는 실시간 음성 방송을 위하여 마이크로 입력되는 음성데이타의 입출력을 담당하는 클래스이다. 이 클래스 또한 mmMediaIOFile이나 mmMediaIOBeeHive와 같이 mmMediaIO 클래스의 인터페이스를 통하여 제어된다.

- mmMediaFile 클래스

이 클래스는 mmMediaIO로부터 얻어진 미디어 스트림이 어떤 형식의 인코딩 방식을 가지고 있는지 판별하고, 각 인코딩 방식에 따라 전송 시 필요한 미디어 정보를 추출하는 역할을 하는 클래스들의 추상화 클래스이다. 현재 AU, WAVE, MPEG-1, MP3로 인코딩되어 있는 미디어와 마이크로 입력되어지는 음성 데이타를 지원하며, CheckMediaFileType() 함수를 이용하여 미디어의 인코딩 방식을 판별한다. 이 클래스는 mmMediaIO 클래스와 같이 향후 추가되는 새로운 형태의 미디어 인코딩 방식을 지원하기 위해서 이 클래스를 상속받고, 추가하고자 하는 미디어의 정보를 추출하는 항목을 작성한 후 mmMediaFile 클래스의 인터페이스를 통하여 제어하면 된다.

- mmMediaFileAu 클래스

mmMediaFile 클래스의 하위 클래스로써 AU 파일의 정보를 추출하고 미디어가 AU 파일 형식인지를 구분하는 클래스이다. 기본적인 함수와 인터페이스는 mmMediaFile

클래스와 같으며, AU 파일헤더 구조체를 지니고 있어 AU 파일의 파일헤더를 읽어 AU 파일이 지니고 있는 정보의 추출이 가능하고, 입력된 미디어가 AU 파일인지를 구분하는 역할을 한다. AU 파일은 저장 방식에 따라서 8비트 u-law, 8비트 linear PCM, 16비트 linear PCM, 24비트 linear PCM, 32비트 linear PCM, 32비트 IEEE floating point, 64비트 IEEE floating point, 8비트 u-law compressed, 8비트 a-law와 같은 여러 가지 형태를 가지고 있으므로 각각의 저장 방식에 알맞은 미디어 정보를 추출한다.

- mmMediaFileWave 클래스

이 클래스는 WAVE 파일의 mmMediaFileAu 클래스와 같은 역할을 한다. 단지 미디어의 인코딩 형식이 WAVE라는 것만이 다르다. 이 클래스는 WAVE의 RIFF 헤더와 chunk 헤더를 위한 구조체를 가지고 있으며, mmMediaFile 클래스의 기본 함수에 chunk부분을 찾는 FindChunk() 함수와 Header를 읽는 ReadHeader() 함수를 추가하여 WAVE 파일의 정보를 추출한다.

- mmMediaFileMpeg1 클래스

이 클래스는 mmMediaFile 클래스를 상속받아 MPEG-1 파일을 지원하기 위한 클래스이다. 이 클래스는 비디오 영역을 다루고 있기 때문에 AU, WAVE, MP3파일보다 좀더 복잡한 형태를 가지고 있다. 이 클래스 또한 다른 mmMediaFile의 하위 클래스들과 같이 기본 클래스에 새로 추가되어지는 함수들을 가지고 있다. 이 클래스는 다른 하위 클래스에 비하여 좀더 많은 함수들을 가지고 있는데 그 이유는 다른 파일보다 복잡한 구조로 이루어져 있기 때문이다. 기본적인 인터페이스 이외에 DetermineStreamType(), ReadMpegInfo(), ReadPackHeader(), ReadSystemHeader(), ReadPacketHeader(), ReadVideoSequenceHeader(), ReadGOPHeader()의 MPEG-1 파일 패킷의 헤더 정보들을 읽어들이는 함수와 MPEG-1 파일의 시작부분과 다음 시작부분을 찾는 함수인 ReadStratCode(), FindNextStartCode()의 함수를 포함하고 있다.

- mmMediaFileMp3 클래스

이 클래스 또한 mmMediaFile 클래스의 하위클래스로 Mpeg Audio Layer 3인 MP3 파일 형식을 지원하는 클래스이다. [표 2]와 [그림 4]는 MP3파일이 가지고 있는 4바이트의 프레임헤더와 ID3 Version TAG의 형식을 보여준다. ID3 Version TAG는 총 128바이트로 구성되어 있는 것으로 MP3파일이 음악의 장르, 가수의 이름 등을 알 수 있는 정보를 지니고 있고 프레임헤더는 비트레이트, 압축방식 등과 같은 MP3파일만의 특성을 가지고 있다. 이 클래스의 전체적인 구성은 mmMediaFile에 선언되어진 기본 함수와 MP3 파일의 헤더와 정보들에 관한 구조체, 프레임헤더를 읽어 들이는 ReadFrameHeader() 함수, ID3 Version TAG를 읽는 과정을 수행하는 ReadId3V1Tag() 함수로 구성되어 있다.

표 2 ID3 Version TAG의 구성

Song Title (up to 30 chars) Artist Name (up to 30 chars) Album Name (up to 30 chars) Year of Recording (up to 4 chars) Comments (up to 30 chars) Genre (one char of 255 possible settings)

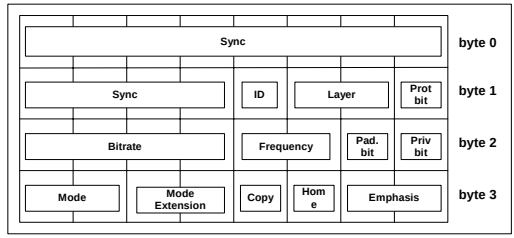

그림 4 Mpeg Audio Layer 3 프레임 헤더

- mmMediaFileLiveS 클래스

이 클래스 또한 mmMediaFile 클래스를 상속받은 클래스로써 마이크로 입력되어지는 음성데이타를 지원하기 위한 클래스이다.

지금까지 미디어 소스 모듈의 클래스들에 관한 간단한 설명을 하였다. [그림 5]와 [그림 6]은 미디어 소스 모듈에서 미디어 소스를 Open하고 Read하는 시퀀스를 나타내는 시퀀스 다이어 그램으로 [그림 5]의 미디어 소스 Open시에는 먼저 CheckMediaFileType() 함수를 호출하여 현재 Open하고자 하는 미디어가 어떤 형식의 미디어인지 체크하고 그 결과 값을 가지고 mmMediaFile의 자식 클래스 중에 하나를 생성하고 mmMediaFile 클래스의 Open() 함수를 호출한다. mmMediaFile 클래스는 다시 미디어 소스와 직접적인 입출력을 담당하는 mmMediaIO 클래스의 Open() 함수를 호출하여 미디어 소스의 Open 과정이 이루어진다. [그림 6]의 미디어 소스 Read는 mmMediaSource 클래스에서 mmMediaFile의 ReadData() 함수를 ReadData() 함수는 mmMediaIO 클래스의 ReadData() 함수를 호출함으로써 미디어의 Read 과정이 이루어진다.

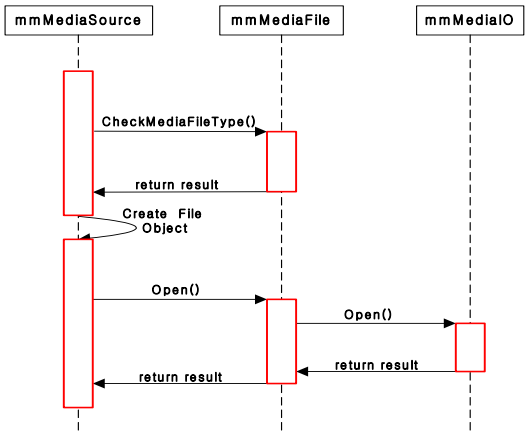

그림 5 미디어 소스 Open 시퀀스 다이어그램

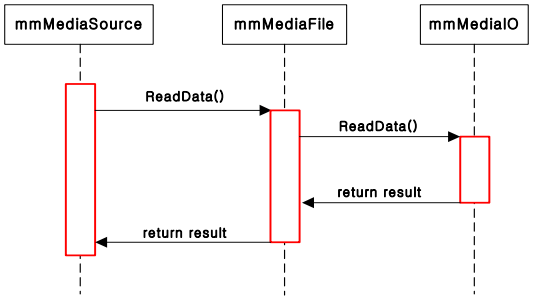

그림 6 미디어 소스 Read 시퀀스 다이어 그램

2) 미디어 싱크 모듈

미디어 싱크 모듈은 전송되어진 미디어 스트림을 재생하는 역할을 하는 모듈이다. 이 모듈은 하드웨어 디코더 보드 등의 영상 디코딩 장치 제어를 위한 mmVideoDev, 다양한 종류의 미디어를 처리하기 위해 DirectShow기술을 이용하는 mmDirectShow, Winamp를 이용한 MP3 멀티캐스팅을 위한 mmWinamp-Gateway, Win32나 Solaris 오디오 데이타의 재생을 위한 mmAudioDev, 전송단과 연계되어 스트리밍의 클라이언트 버퍼링을 관리하는 mmBufferMgr 클래스로 구성되어 있으며 mmMediaSink 클래스를 통하여 각 클래스들이 사용되어 진다.

- mmVideoDev 클래스

하드웨어 디코더 보드 등과 같은 하드웨어 비디오 장치 제어를 위한 클래스로 미디어 싱크에 전송된 미디어 포맷이 MPEG-2인 경우에 이 모듈을 통해서 디코딩된다. 이 클래스는 DirectShow가 기본적으로 제공하지 못하는 형태의 포맷인 MPEG-2를 하드웨어 디코더 보드를 이용해서 처리할 수 있게 하는 선택적인 클래스이다. 현재 MPEG-2는 국내기업인 바로비젼사에서 개발한 소프트웨어 MPEG-2 디코더인 DirectShow기술의 트랜스폼 필터를 이용하여 MPEG-2를 재생할 수 있다.

- mmDirectShow 클래스

이 클래스는 미디어 싱크 모듈에서 다양한 미디어의 형식을 제공하기 위해 마이크로소프트사의 DirectShow 기술을 본 연구에서 개발한 미디어 관리자의 구조에 맞게 적용되어진 클래스이다. 제안된 미디어 관리자에서는 UDP기반의 RTP/RTCP/RTSP 프로토콜을 지원할 수 있고 DirectShow의 기준을 따르고 ISSA 프레임워크에 알맞은 형태의 RTP 소스 필터를 새로 개발하여 사용하였다. DirectShow의 구조는 [그림 7]과 같으며 미디어를 읽어들이는 소스 필터와 미디어 데이타를 알맞은 소프트웨어 디코더와 연결하여 미디어를 디코딩하여 디바이스에서 디스플레이하기에 적당한 형태로 변환해주는 트랜스폼 필터, 그리고 변환된 데이타를 디바이스로 출력하는 랜더러 필터로 구성된다[8].

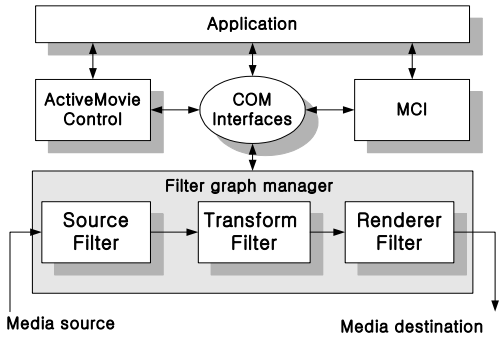

그림 7 DirectShow의 구조

개발된 소스필터는 UDP기반의 RTP 프로토콜을 사용할 수 있고, ISSA 프레임워크를 이용한 미디어 스트리밍 서비스의 클라이언트로 쓰일 수 있는 미디어 플레이어의 개발이 매우 쉬워졌을 뿐만 아니라, 타 회사 또는 타 연구에서 DirectShow기술을 이용하여 개발된 미디어 플레이어에 본 연구에서 개발된 RTP 소스 필터로 대체만 하면 그 미디어 플레이어들을 본 미디어 관리자에 이용할 수 있다. 본 논문에서 다루는 미디어 관리자에서는 마이크로소프트사의 Windows Media Player에 개발한 RTP 소스 필터를 적용시켜 VOD 스트리밍 서버를 구축하였고, 바로비젼사에서 개발된 MPEG-2 소

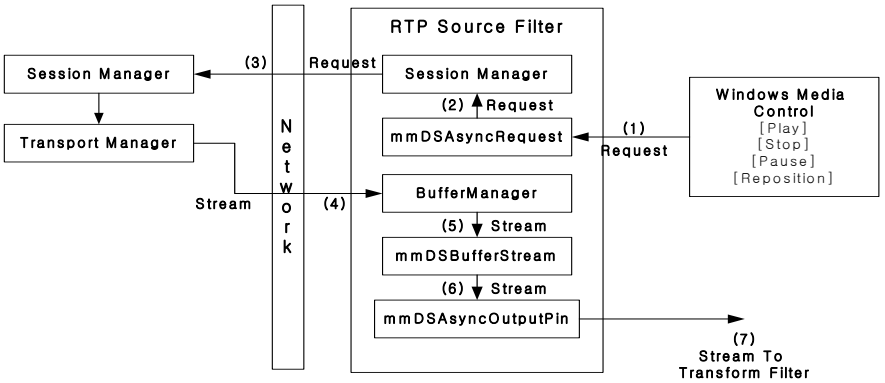

그림 8 RTP 소스 필터의 내부고조 및 동작 순서

프트웨어 트랜스폼 필터를 본 RTP 소스 필터와 함께 사용하여 MPEG-2의 스트림을 하드웨어 디코더의 추가 장착이 없이 Windows Media Player로 처리할 수 있다. 물론 MPEG-2 하드웨어 디코더 보드가 Direct Show의 필터구조를 지원하는 API를 제공한다면 이것 역시 개발된 RTP 소스 필터와 함께 이용될 수 있다. 또 다른 응용으로는 바로비젼사에서 개발한 VaroDVD라는 미디어 플레이어에서 사용되어지는 소스 필터를 본 연구에서 개발한 RTP 소스 필터로 대체하면 VaroDVD라는 소프트웨어 DVD 플레이어에서도 RTP 스트리밍을 가능하게 할 수 있다. 이러한 점이 본 논문에서 제안한 미디어 관리자가 다른 미디어 관리자와 차별화 되는 강력한 확장성과 응용성을 가지는 이유이다.

- RTP 소스 필터(Source filter)

소스필터(Source filter)의 역할은 로컬 디스크나 네트워크 또는 인터넷으로부터 데이타를 가져와 트랜스폼필터(Transform filter)로 전달 역할을 수행하는 Direct Show의 COM 객체이다. [그림 8]은 ISSA 프레임워크에서 개발된 RTP 소스필터의 내부구조 및 작동 순서를 보여주는데, 각 객체를 이어주는 화살표가 작동순서를 나타낸다.

Windows Media Control 객체에서 PLAY, STOP, PAUSE, REPOSITION 등과 같은 사용자 요구는 mmDSAsyncRequest 객체로 보내지며, 이 요구사항은 세션 관리자를 통하여 서버의 세션 관리자에게 전달된 후 전송 관리자를 통해 미디어 스트림을 전송하고, 전송이 시작되면 버퍼관리자의 버퍼링을 통하여 소스필터의 버퍼를 담당하는 mmDSBufferStream 객체로 미디어 스트림을 전달하게 된다. 이렇게 전달되어진 미디어 스트림은 소스필터의 출구인 mmDSAsyncOutputPin을 통해서 다음에 단계에 있는 DirectShow의 트랜스폼필터에게 전달한다. 트랜스폼필터에 의해 디코딩되어진 미디어 데이타는 디스플레이 디바이스와 오디오 디바이스를 통하여 재생된다. 즉 미디어 타입을 선택한 소스필터에 의해 미디어 타입에 알맞게 필터 그래프가 구성되고 그 필터그래프를 통하여 재생이 되어지는 것이다.

[그림 9]는 RTP 소스 필터의 동작 알고리즘을 보여준다. 현재 RTP 소스 필터에서 미디어 스트림에 관련된 사용자의 요구 종류는 PLAY, STOP, PAUSE의 3가지가 있다. PLAY는 미디어 재생을 위한 요구이고, STOP은 미디어 재생을 끝마치고 미디어 전송에 관련

```
If there are user's requirements {
    Switch requirements
    Case PLAY:
        send a requirement to the client session;
        send a requirement to the client I/O Object;
        initialize client's I/O operation;
        If connection is established to the server session
            and success to initialization {
            initialize the transport manager;
            the server sends a media stream to the client;
            after doing buffering, the client sends
                a stream to the transform-filter;
        }
        Else {
                the server stops a stream sending;
                disconnect the client/server session;
                initialize buffer;
        }
    Case STOP:
        the server stops a stream sending;
        disconnect the client/server session;
        initialize buffer;
    Case PAUSE:
        the server is pending to send a stream;
        keep the server/client's session connection;
        standby for the next user's requirement;
}
```

그림 9 RTP 소스 필터의 동작 알고리즘

된 정보를 가진 서버와의 세션이 더 이상 필요하지 않을 때 또는 사용자가 미디어 전송을 중단하려 할 때 요구이며, PAUSE는 미디어 전송에 관련된 정보를 가진 서버 세션과의 연결은 유지한 채 잠시 미디어 재생을 멈춘 상태이다. PAUSE 명령을 요구하면 PLAY 명령의 수행중인 경우에는 미디어의 전송이 잠시 멈추게 되며 다시 PAUSE 명령을 내리면 멈춘 지점부터 다시 재생이 된다. 그리고 다음 요구가 STOP인 경우에는 재생을 마치고 세션과 연결을 끊게 된다.

전송 관점에서 볼 때, PLAY 상태의 경우에는 서버 측에서 미디어 포맷에 따라 가장 적절한 크기의 패킷을 사용해 미디어 스트림을 클라이언트로 보낸다. STOP의 경우에는 서버 측에서 미디어 스트림 전송을 중단하고, 버퍼를 초기화 한 후, 세션 정보를 삭제한다. PAUSE의 경우에는 미디어 스트림 전송은 중단하지만, 버퍼와 세션 정보는 그대로 유지한 채 클라이언트로부터 다음 요구를 기다리는 상태가 된다. RTP 프로토콜을 이용하고 버퍼 관리자를 통해 클라이언트로 전송된 미디어 스트림 패킷은 조립과정을 거쳐 미디어 스트림으로 환원이 되고 이것은 RTP 소스필터의 Output 핀을 지나서 디코딩을 위한 트랜스폼필터로 이동하게 된다.

[그림 10]은 RTP 소스필터의 클래스 다이어그램을 보여준다. mmDSAsyncOutputPin클래스는 모든 핀의 기본이 되는 CBasePin과 비동기 입출력 작업의 기본이 되는 IAsyncReader 클래스를 상속받아 생성된다. 실질적인 입출력 작업을 담당하는 mmDSAsyncIO 클래스는 현재 스트리밍 되고 있는 미디어의 위치에 대한 정보를 요구하는 mmDSAsyncRequest와 RTP 프로토콜 처리부분을 담당하는 mmDSAsyncStream과 함께 연관되어 작동한다. mmDSBufferStream 클래스는 mmDS Async Stream을 상속받아 생성하며, mmDSBuffer Reader 클래스는 CBaseFilter를 상속받은 클래스인 mmDSAsyncReader 클래스를 상속받아 생성하고 mmDSAsyncOutputPin 클래스를 이용한다. RTP 소스필터는 미디어의 입출력을 담당하는 mmDSBuffer Stream 클래스와 소스필터를 구성하는 mmDSBuffer Reader 클래스를 이용하여 만들어진다.

RTP 소스필터의 개발로 ISSA 프레임워크의 클라이언트는 DirectShow의 필터들을 이용하여 개발된 미디어 플레이어들에 쉽게 적용시킬 수 있어 사용자들은 취향에 맞는 다양한 클라이언트를 선택할 수 있게 되었다.

- mmWinampGateway 클래스

mmWinampGateway 클래스는 ISSA 프레임워크를 이용한 MP3 멀티캐스트 방송용 클라이언트로 Winamp를 사용하기 위한 Winamp Gateway 플러그인의 인터페이스를 가진 클래스로써 본 미디어 관리자에서 다양한 클라이언트 환경을 가질 수 있는 확장성을 위하여 구현되었다. 개발된 Winamp Gateway 플러그인은 멀티캐스트 방송 서버에서 스트리밍 되어진 미디어 데이타를 Winamp를 통하여 재생시킨다. Winamp는 다양한 형태의 미디어와 미디어 재생방식을 지원하고 많은 형태의 플러그인이 존재하는 대표적인 오디오 재생 전용 소프트웨어이다. Winamp의 전체적인 구조는 재생방식과 부가적인 기능을 가지고 있는 플러그인, 로컬이나 원격에서 전송되어진 미디어 데이타를 디코딩 해주는 오디오 Codec부분, Winamp의 윈도우 스킨과 아이콘 등의 설정을 정해주는 Display부분, 디코딩 되어진 미디어 데이타를 재생시켜주는 오디오 재생 부분으로 구성되어 있다. Winamp에서 지원 가능한 플러그인은 Winamp 개발자들이 정의한 인터페이스에 따라 구현하는 방법과

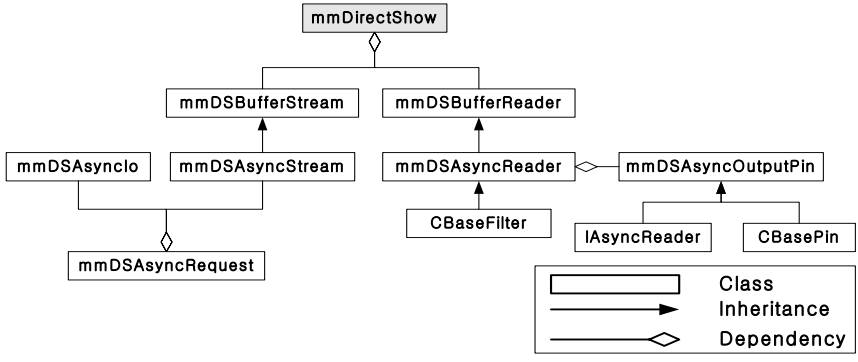

그림 10 RTP 소스필터의 클래스 다이어그램

Win32 Message API를 이용한 메시지 전달 방법으로 자신이 원하는 플러그인을 손쉽게 만들 수 있다[14].

• Winamp Gateway 구조 및 설계

Winamp Gateway init() 함수와 quit() 함수로 이루어져 있다. init() 함수는 ISSA의 미디어 관리자의 미디어 싱크 모듈을 이용하여 오디오 데이타 패킷을 받아 초기화 과정을 수행하는 함수로, 미디어 싱크 모듈의 Open() 함수와 Winamp를 제어하는 부분으로 구분된다. quit() 함수는 플러그인을 종료하기 위해 서버와 연결된 세션을 종료하는 역할을 한다. init() 함수를 통하여 미디어 싱크 모듈의 Open() 함수를 호출하면 멀티캐스트 서버에서 전송되어진 패킷을 전송 받기 전에 Winamp에게 전송할 TCP 포트를 열어주고, 멀티캐스트 서버에서 오디오 데이타 패킷 전송이 시작되면 미디어 관리자의 버퍼링 모듈을 통해 버퍼링이 시작되고, TCP 포트를 통하여 Winamp에게 오디오 데이타가 전송이 시작되게 된다. 작동 알고리즘은 [그림 11]과 같다.

```
Algorithm for Winamp Gateway Operation

    create the TCP transport port
    buffering the RTP transport data
    send the Audio data to the Winamp
    play the Winamp Audio data
```

그림 11 Winamp Gateway 작동 알고리즘의 의사 코드

[그림 12]은 Winamp Gateway의 작동에 사용되어지는 클래스들의 클래스 다이어 그램으로 ISSA 프레임워크의 NetInterface모듈을 이용한다. NetInterface 모듈은 NetByteOrder, NetAddress를 상속받은 NetInetAddr, 그리고 네트워크 인터페이스가 정의되어진 NetSocket의 상속을 받은 NetSocketTCP와 NetSocketUDP클래스가 있으며, 멀티 캐스트를 위해 NetSocketUDP를 상속받은 NetSocketMcast가 있다. mmWinampGateway 클래스는 TCP 전송 포트를 생성하기 위하여 NetInterface의 NetSocketTCP클래스를 이용한다.

• mmAudioDev 클래스

이 클래스는 Win32 오디오 또는 Solaris 오디오 장치 등과 같이 오디오 데이타 표현을 위한 클래스로 특별한 압축방식을 가지지 않는 오디오 데이타를 처리하기 위해서 사용된다.

3) Push/Pull을 동시에 지원하는 버퍼관리자

본 논문에서 다루는 Push/Pull 버퍼 관리 기법은 클라이언트측의 mmMediaSink 클래스에서 동작하며 서버로부터 전송되는 미디어 스트림 데이타를 버퍼링 하고, 버퍼에 있는 데이타를 Push 및 Pull 방식으로 미디어 재생 장치로 전달하는 기능을 하나의 구조에서 제공한다[15].

• 버퍼 관리자 모델

클라이언트에서의 Push/Pull 버퍼 관리 기법의 전체적인 동작 구조는 [그림 13]과 같이 이루어지며, 실제 클라이언트에 위치한 Push/Pull 버퍼 관리 기법이 미디어 재생 장치와 연관되어 동작하는 구조를 표현한 것이다.

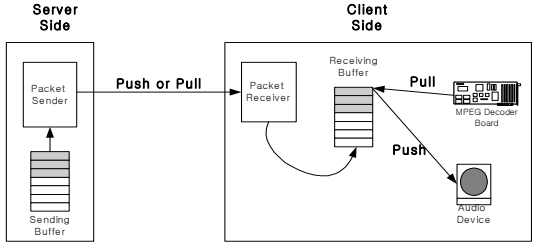

그림 13 Push/Pull 버퍼 관리 기법의 동작 구조

[그림 13]에서 서버 측의 패킷 송신자를 통해 Push나 Pull 방식으로 전송된 패킷은 클라이언트에서의 패킷 수신자를 통해 수신되어 수신 버퍼로 전달된다. 수신 버

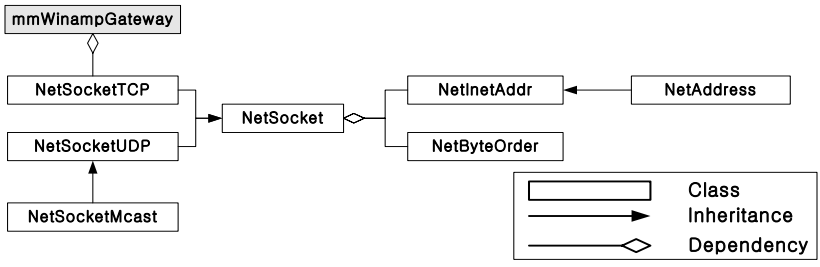

그림 12 Winamp Plug-in 클래스 다이어 그램

퍼로 전달된 미디어 스트림 데이타는 설정된 미디어 재생 장치로 초기에 설정된 Push 혹은 Pull의 데이타 전달 방식을 이용해 재생된다. 즉 버퍼 관리 기법은 초기에 교섭을 통해 미디어 표현장치에 알맞은 데이타 전달 방식을 설정한 후 전송 세션이 시작하면 이미 설정된 전달 방식에 따라 데이타를 적절한 장치로 전달하며, 이때 미디어 재생 장치는 설정된 데이타 전달 방식을 지원할 수 있는 장치이다. 또한 이때 버퍼로 데이타를 입력해주는 *Packet Receiver*는 서버로부터 데이타를 Push 방식으로 받을 수도 있으며 Pull 데이타 요청을 처리하는 stub을 설치하면 Pull 방식으로도 데이타를 받을 수 있어, 실제 수신 버퍼는 입력 데이타의 네트워크 전달 방식과 상관없이 유연하게 동작한다.

5. 미디어 관리자의 구현

본 논문에서 설계한 미디어 관리자는 ISSA 프레임워크의 일부분으로써 ISSA 프레임워크를 이용한 스트리밍 응용 소프트웨어 개발하는데 중요한 역할을 하는 부분으로 스트리밍 응용 소프트웨어의 한 종류인 VOD/AOD 서비스의 구현시 서버와 클라이언트 구현에서 전송 시 필요한 정보를 얻고 다양한 종류의 재생방식을 채택할 수 있는 환경을 제공하도록 구현하였다. [그림 14]는 미디어 관리자를 이용한 간단한 VOD/AOD 스트리밍 서버, 클라이언트를 구현했을 때 미디어 관리자의 구현관점에서 본 논리적인 구조를 나타낸다.

미디어 관리자의 가장 큰 두 가지 모듈인 미디어 소스와 미디어 싱크 모듈을 중심으로 전체적인 구조를 설명할 수 있다. 먼저 미디어 소스 모듈은 스트리밍 서버에서 세션 관리자, 파일 시스템, 실시간 멀티미디어 데이타베이스인 BeeHive와 연동을 위한 데이타베이스 커넥터와 연결되어 웹 브라우져를 통해 접수되는 사용자의 미디어 컨텐츠에 대한 요구가 웹서버를 통해 AMS (Agent for Multimedia Serveice) 서버로 전달되어 스트리밍 서버 에 선택되어진 미디어 컨텐츠를 클라이언트로 전송할 것을 명령하고, 미디어 소스 모듈은 선택된 미디어 컨텐츠를 데이타베이스나 파일 시스템으로부터 미디어 스트림을 얻어와 전송 관리자에게 연결시키고 전송관리자는 미디어 타입에 맞게 패킷을 만들어서 클라이언트로 전송한다. 다음으로 미디어 싱크 모듈은 전송관리자로부터 전송되어진 미디어 스트림 데이타가 버퍼관리자와 연결되고 버퍼관리자는 전송된 미디어 데이타의 버퍼링을 하고 미디어 싱크 모듈은 전송되어진 미디어 타입에 따라서 DirectShow 기술을 사용한 ISSA 프레임워크에 적합하도록 구현되어진 RTP 소스필터를 이용하거나, Winamp등의 가장 알맞은 미디어의 재생 모듈과 연결 미디어를 재생하게 된다.

미디어 소스와 미디어 싱크 모듈은 각각의 하위 모듈에 접근하기 쉽게 하위 모듈들과 동일한 인터페이스를 구성하는 방법으로 추상화를 시켰다. 기본적인 인터페이스의 메소드는 [표 3]에 잘 나타나 있다. [표 3]에서 볼 수 있듯이 미디어 소스 모듈은 미디어를 저장 매체로부터 읽어오는 작업을 미디어 싱크 모듈은 전송되어진 미디어를 재생시키는 작업을 주로 한다는 것을 알 수 있다.

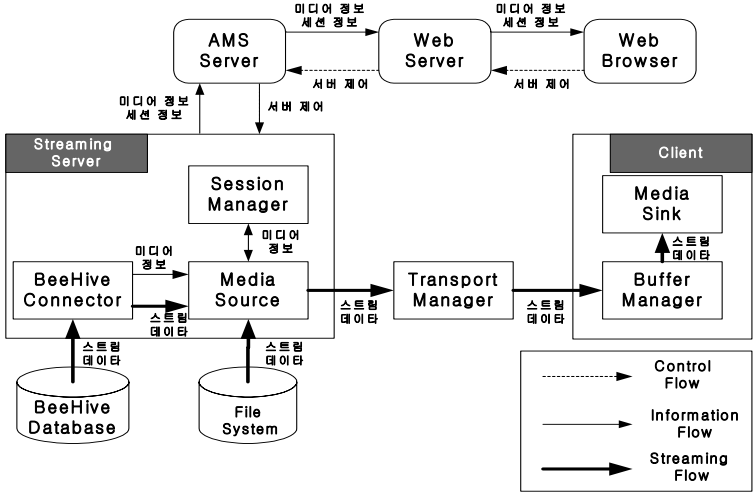

그림 14 미디어 관리자 관점에서 본 VOD/AOD 서비스의 논리적 구조

표 3 미디어 소스와 미디어 싱크의 기본 인터페이스

미디어 소스 모듈의 기본 메소드	Open(), Close(), ReadData()
미디어 싱크 모듈의 기본 메소드	Open(), Close(), WriteData()

다음은 본 논문에서 구현한 미디어 관리자를 이용하여 개발할 수 있는 응용 분야의 하나의 예로써 VOD/AOD 시스템의 구성과 구현결과를 보여준다.

5.1 RTP 소스 필터를 이용한 VOD 서비스

미디어 관리자를 이용한 VOD 스트리밍 서버는 홈페이지 서버와 실질적인 미디어 소스를 가지는 파일서버 또는 실시간 멀티미디어 데이타베이스 시스템으로 구성된다. 홈페이지 서버는 사용자 인터페이스로서, 사용자가 재생을 원하는 미디어 파일의 사진이나 이름을 선택하는 것으로 스트리밍 서비스를 받을 수 있고, 클라이언트는 마이크로소프트사의 Windows Media Player를 이용한다. ISSA 프레임워크에서 클라이언트로 Windows Media Player를 이용하기 위해서는 ISSA 프레임워크에 알맞고 미디어 관리자의 관리를 받을 수 있도록 제작되어진 RTP 소스필터를 이용한다. 이 시스템의 작동 순서는 [그림 15]와 같다. 먼저 사용자가 원하는 동영상을 선택하기 위해 홈페이지 접속하게 된다. 홈페이지를 통하여 선택되어진 미디어 파일은, 그것이 링크된 ASX 파일을 활성화시킨다. 활성화된 ASX 파일은 스트리밍 서버와 연결을 시도하고 스트리밍이 시작되면 RTP 소스 필터가 작동하고 [그림 16]과 같이 Windows Media Player가 실행되어 선택된 미디어를 볼 수 있다.

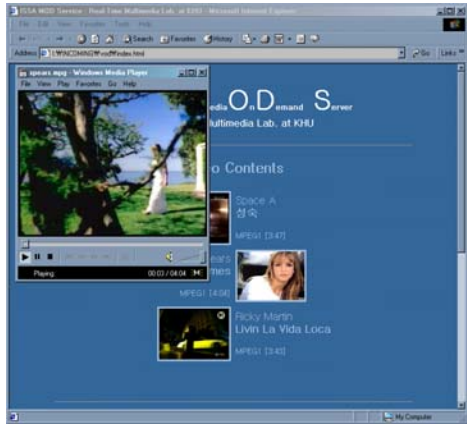

그림 16 Window Media Player 구현 화면

ASX 파일은 메타 파일로서 사용되며 사용되어지는 프로토콜이나 미디어 파일에 대한 정보를 유지한다. 그 역할은 사용자의 컴퓨터 내에 있는 Windows Media Player를 활성화시키고, 실질적인 미디어 파일을 가지는 스트리밍 서버에 미디어의 스트리밍을 시작하도록 해준다. 이런 접근을 통해서 각 서버의 작업량을 분산시킬 수 있고, 사용자에게 별도의 추가작업이 없이 자동으로 미디어 플레이어를 활성화시킬 수 있는 편의성을 제공하도록 구현할 수 있다.

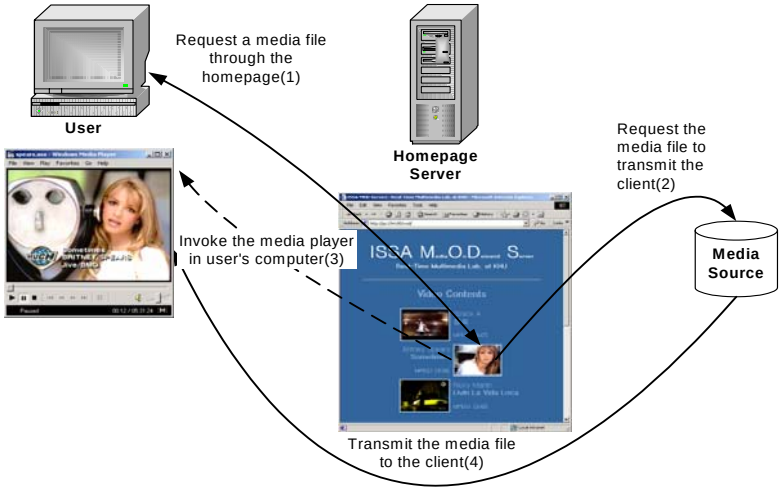

그림 15 ISSA VOD 서비스 구현 순서도

5.2 Winamp Plug-in을 이용한 AOD 서비스

미디어 관리자를 이용한 AOD 서버는 멀티캐스트를 지원하고, 클라이언트로 Winamp를 이용한다. Winamp를 이용하기 위해서는 Winamp Gateway라고 하는 본 연구에서 구현한 Winamp용 Plug-in을 이용한다. Winamp를 이용한 멀티캐스트 AOD 서비스의 작동 순서는 [그림 17]과 같다. 사용자가 AOD 서비스 홈페이지 서버에 접속을 해서 자신이 듣고 싶어하는 미디어를 선택하면, 선택된 미디어는 멀티캐스트 AOD서버에게 전달되어 멀티캐스트 미디어 리스트에 추가되어 진다. 그리고 사용자가 음악을 선택하는 동시에 사용자 컴퓨터의 Winamp 프로그램을 실행하고 멀티캐스트 서버의 IP-Address와 채널 정보를 메타파일을 통하여 전달한다. Winamp는 [그림 18]과 같이 홈페이지를 통하여 얻어진 멀티캐스트 서버의 정보를 이용하여 멀티캐스트 서버와 접속을 하게 되며, Winamp Gateway를 이용하여 멀티캐스트 서버로부터 패킷을 전송을 받고 재생이 이루어지고, 현재 방송되어지는 미디어를 감상할 수 있게된다. 사용자가 선택한 미디어 또한 스케줄러에 의해 정해진 일정한 순서가 되면 들을 수 있게 된다. Winamp Plug-in을 이용한 AOD용 클라이언트를 제작함으로써 ISSA 프레임워크의 미디어 관리자를 이용한 스트리밍 시스템을 구축하는데 있어 클라이언트를 구현할 필요없이 간단한 Plug-in의 구현으로 가능하다는 것을 알 수 있다.

그림 18 Winamp의 작동 구현 화면

6. 결론 및 향후연구

본 논문에서는 멀티미디어 스트리밍 프레임워크인 ISSA에서 미디어 관리자의 설계와 구현에 대한 경험을

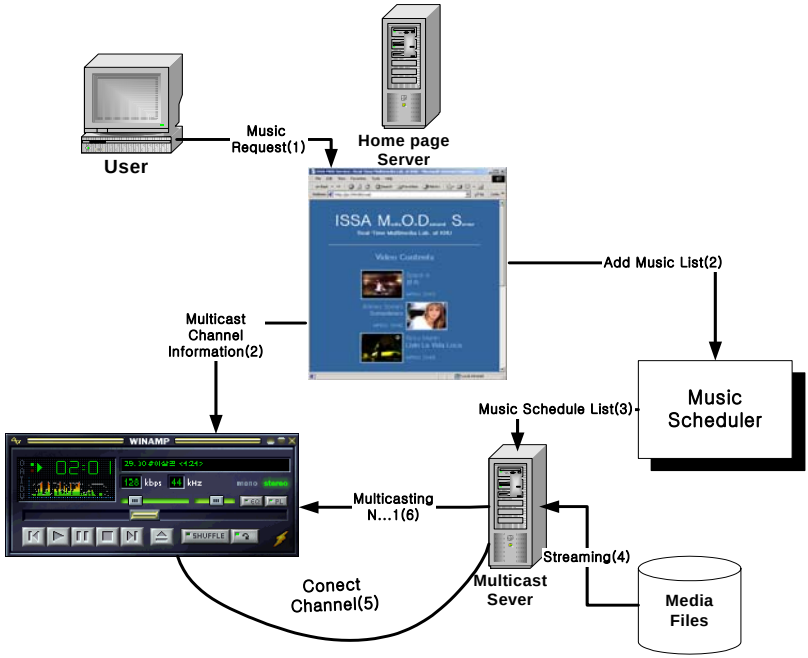

그림 17 Winamp Gateway을 이용한 AOD 서비스 구현 순서도

기술하였다. ISSA에서 미디어 관리자는 로컬 파일이나, 데이타베이스에서 관리되는 미디어 스트림, 실시간 동영상 시스템에서 얻어온 미디어 스트림들의 추상화를 통하여 일관성 있는 미디어 소스의 처리와 클라이언트에서의 미디어 처리가 간단해지는 것이 가능하였고, 실시간 멀티미디어 데이타베이스인 BeeHive와 연동이 가능하도록 구성된 미디어 관리자를 통하여 데이타베이스의 효율적인 관리를 받는 미디어들을 스트리밍 할 수 있다. 또한, 클라이언트에서의 소프트웨어 디코딩을 위하여 채택한 DirectShow 기술을 이용함으로써 편리하고 강력한 디코딩을 지원하는 코덱을 제공하고, 범용성 및 유연성과 확장성을 갖추고 있어 향후에 나타날 새로운 형태의 미디어 포맷에 강력하게 대처할 수 있음을 보여준다. 또한 서버의 환경이 WindowNT나 Solaris의 하나에만 국한된 것이 아니라 다양한 운영체제에서 활용이 가능하다.

그리고, 다양한 미디어 재생 장치로 데이타 공급을 위해 단일 메모리에서 Push 방식과 Pull 방식의 버퍼관리 기법을 모두 지원할 수 있고, 네트워크 지터의 적응을 위해 버퍼링 기능을 제공할 수 있는 효율적이고 유연한 클라이언트에서의 Push/Pull 버퍼 관리 기법을 개발하였다. 이는, 다양한 미디어 재생장치의 각기 다른 버퍼 관리 요구를 단일 메모리에서 일관되게 처리함으로써 메모리 낭비를 방지할 수 있으며, 융통성 있고 단순한 버퍼관리가 가능하게 되었다.

향후 연구로는, DirectShow 기술의 적용으로 클라이언트로 마이크로소프트사의 Windows Media Player를 이용하기 때문에 우수한 성능과 다양한 미디어의 포맷을 지원할 수 있는 유연성을 얻었지만 스트리밍 서버의 성능향상을 위해 버퍼링 시간과 버퍼 크기의 조절 등 미디어 관리자 내의 버퍼관리자 기능을 개선할 예정이다. 그리고, DirectShow가 기반으로 두고 있는 COM 기술에 대한 보다 깊은 연구를 통해 견고하고 안정적인 소스필터를 구현하고, 다양한 미디어 소스와 클라이언트를 이용할 수 있도록 더 많은 확장 모듈들을 구현 할 것이며, 실시간 멀티미디어 데이타베이스인 BeeHive와의 연동 시 보다 처리시간이 빠르고 안정적인 스트리밍을 위한 미디어 관리자 내부의 모듈에 대한 개선을 할 것이다.

참 고 문 헌

[1] K. Mayer-Patel, and L. A. Rowe, "Design and Performance of the Berkeley Continuous Media Toolkit," In Multimedia Computing and Networking 1997, In Proc. of SPIE 3020, pp.194-206, 1997.

[2] C. J. Lindblad, and D. L. Tennenhouse, "The VuSystem: A Programming System for Compute-Intensive Multimedia," In IEEE Journal of Selected Areas in Communications, 1996.

[3] 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.

[4] C.G. Jeong, H.I. Kim, Y.R. Hong, E.J. Lim, S. Lee, J.W. Lee, B.S. Jeong, D.Y. Suh, K.D. Kang, John A. Stankovic, and Sang H. Son, "Design for an Integrated Streaming Framework," Department of Computer Science, University of Virginia Technical Report, CS-99-30, November, 1999.

[5] H. Schulzrinne, "RTP Profile for Audio and Video Conferences with Minimal Control," IETF RFC 1890, Jan. 1996.

[6] H. Schulzrinne, "RTSP: Real-Time Streaming Protocol," IETF RFC 2326, Apr. 1998.

[7] J. Stankovic and S. H. Son, "An Architecture and Object Model for Distributed Object-Oriented Real-Time Databases," *Journal on Computer Systems Science and Engineering*, Special Issue on Object-Oriented Real-Time Distributed Systems, vol. 14, no. 4, pp.251-259, July 1999.

[8] DirectShow를 이용한 개발 참고 사이트, http://www.microsoft.com/DirectX/dxm/help/ds/

[9] JMF(Java Media Framework) 참고 사이트, http://java.sun.com/marketing/collateral/jmf.html

[10] 이종민, 차호정, "웹 기반의 VOD 시스템 구현", 1997년 한국정보처리학회 추계 학술발표논문집 제4권 제2호, pp.50-53, 1997년 11월.

[11] 김형일, 이승룡, "멀티미디어 QoS를 위한 미디어 객체 구조의 설계", '98 한국 정보과학회 춘계 학술발표 논문집, pp.699-701, 1998년 4월.

[12] 스트리밍 시스템 참고 사이트, http://www.publicsource.apple.com/projects/streaming/

[13] 상용 스트리밍 서버 제작사, http://streaming.entera.com/

[14] Winamp Plug-in 개발 참고 사이트, http://www.winamp.com/nsdn/

[15] 정찬균, 이승룡, "멀티미디어 통신시스템을 위한 클라이언트에서의 Push/Pull 버퍼관리기법", 정보처리학회 멀티미디어 특집 논문집, pp.721-732, 2000년 3월.

이 재 욱

2000년 한신대학교 정보통신학과 졸업. 2001년 현재 경희대학교 전자계산공학과 석사과정 재학중. 관심분야는 멀티미디어 스트리밍

이 승 룡

1978년 고려대학교 재료공학과 졸업. 1986년 Illinois Institute of Technology 전산학과 석사. 1991년 Illinois Institute of Technology 전산학과 박사. 1992년 ~ 1993년 Governors State University 조교수. 1993년 ~ 현재 경희대학교 전자계산공학과 부교수. 관심분야는 실시간 컴퓨팅, 실시간 미들웨어, 멀티미디어 시스템, 시스템 보안

홍 인 기

1989년 연세대학교 전기공학과 졸업. 1991년 연세대학교 전기공학과, 통신시스템 공학 석사. 1995년 연세대학교 전기공학과, 통신시스템 공학 박사. 1995년 ~ 1999년 SK Telecom IMT-2000 추진본부 선임연구원. 1999년 ~ 현재 경희대학교 전자정보학부 조교수. 관심분야는 이동통신, CDMA, IMT-2000