논문 01-26-6A-13 한국통신학회논문지 '01-6 Vol.26 No.6A

통합 스트리밍 서비스 구조 설계 및 구현

정회원 이 승 룡*, 홍 충 선**

Design and Implementation of an Integrated Streaming Service Architecture

Sungyoung Lee* Choong Seon Hong** *Regular Members*

요 약

본 논문에서는 멀티미디어 스트리밍을 위한 표준 프로토콜과 미디어 포맷을 지원하고, 실시간 객체지향형 멀티미디어 데이터베이스(BeeHive)를 지원하는 통합 멀티미디어 스트리밍 서비스 프레임워크(ISSA)를 제안한다. ISSA의 핵심 구성요소는 세션의 생성과 스트림 제어를 담당하는 세션 관리자, 스트리밍 데이터의 전송을 담당하는 전송 관리자, 미디어의 압축과 복원 및 재생을 담당하는 미디어 관리자, 그리고 데이터베이스와의 연동을 담당하는 데이터베이스 커넥터이다. ISSA는 스트림의 제어를 위하여 IETF에서 제안한 RTSP, 스트림과 세션의 표현과 초기화를 위해 SDP와 SIP, 스트림 데이터의 실시간 전송을 위하여 RTP 및 TCP, UDP를 지원한다. 또한, ISSA는 VOD, MOD, 웹캐스팅을 위한 개발자 라이브러리를 제공하며, 웹서버와 CORBA A/V 스트리밍 서비스와의 연동을 위한 게이트웨이도 제공한다. ISSA는 운영체제 환경에서 플랫폼에 독립적으로 작동하고, 객체지향 멀티미디어 데이터베이스 서비스를 지원한다. 그리고 멀티캐스팅, 웹캐스팅, 멀티미디어 통신을 위한 표준 프로토콜을 지원하며, 상용 코덱인 Microsoft 사의 Direct Show를 지원할 수 있는 RTP 소스필터를 가지고 있다. 마지막으로, 기존의 다양한 표준 미디어 포맷을 지원할 뿐만 아니라, 새로운 미디어 포맷도 쉽게 수용할 수 있도록 설계되어 있다.

ABSTRACT

In this paper, we propose an Integrated Streaming Service Architecture (ISSA) which supports various types of media format, standard communication protocols, and interconnecting method between the ISSA and the BeeHive, an object-oriented real-time multimedia database developed by the University of Virginia. The core components of the ISSA are the session manger dealing with creation and control of session, the transport manager taking charge of data transmission, the media manager handling to coding and decoding data, and the BeeHive connector interconnecting between ISSA and the BeeHive. The ISSA can support various type of standard communication protocols such as RTSP, SDP, SIP, RTP over TCP and UDP. It also provides a set of development library functions for VOD, MOD and webcasting, and the CORBA gateway module for interconnecting between the CORBA A/V streaming service and the ISSA. Some distinguishable features of the ISSA are platform independent, service for the object-oriented real-time multimedia databases, multicasting as well as web casting, diverse protocols for standard multimedia communications, and the source filter for Microsoft's Directshow. It also supports many existing media formats and has a capability to adopt any types of new comming media formats.

Ⅰ. 서 론

스트리밍 기술은 기존의 파일 다운로드 방식과 달리 네트워크 상에서 오디오와 비디오, 애니메이션, MIDI 등의 멀티미디어 데이터를 실시간으로 송수신하면서 동시에 재생하는 기술이다. 현재 스트리밍

* 경희대학교 전자정보학부 (sylee@oslab.kyunghee.ac.kr), ** 경희대학교 전자정보학부 (cshong@khu.ac.kr)
논문번호: K01126-0508, 접수일자: 2001년 5월 8일

시장을 선점하고 있는 제품으로는 RealNetwork 사의 RealSystem G2[1], Microsoft 사의 Windows Media Technology[2], Apple 사의 QuickTime[3] 등이 있으며, 이외에 다양한 제품들이 스트리밍 서비스를 지원하고 있다.

RealSystem G2는 제어 프로토콜과 전송 프로토콜로 IETF의 RTSP(Real-Time Streaming Protocol)[4]와 RTP(Real-Time Transport Protocol)[5] 등의 표준 프로토콜을 사용하지만, 스트리밍 미디어 포맷은 RealAudio, RealVideo, RealPix 등 자사의 독특한 미디어 포맷을 사용하여 스트리밍을 지원하고 있다. Windows Media 또한 자사의 MMS(Multi-Media Streaming) 프로토콜을 사용하여 스트림을 제어하고 있으며, ASF(Advanced Streaming Format)와 Windows Audio Format 등 자사 고유의 스트리밍 포맷을 사용하고 있다. QuickTime은 제어 프로토콜로서 IETF의 RTSP를 사용하고 있지만, 스트리밍 포맷은 역시 자사의 QuickTime 포맷을 사용하고 있다. 이러한 스트리밍 제품들은 자사 고유의 프로토콜과 스트리밍 미디어 포맷을 사용함으로써 상호 운영성이 부족하며, 이식성과 유연성, 확장성이 결여되어 있다. 또한 파일시스템 및 관계형 데이터베이스 시스템을 사용함으로써 효율적인 멀티미디어 데이터 관리 및 멀티미디어 데이터의 검색과 추출이 어렵다.

본 논문에서는 멀티미디어 스트리밍을 위한 표준 프로토콜과 미디어 포맷을 지원하고, 멀티미디어 데이터의 효율적인 관리를 위하여 객체지향형 실시간 멀티미디어 데이터베이스 시스템인 BeeHive[6]를 연동한 통합 멀티미디어 스트리밍 서비스 프레임워크(Integrated Streaming Service Architecture: ISSA)를 제안한다. ISSA는 핵심 구성요소는 세션의 생성과 스트림 제어를 담당하는 세션 관리자(Seesion manager)[7], 스트리밍 데이터의 전송을 담당하는 전송 관리자(Transport manager)[7], 미디어의 압축과 복원 및 재생을 담당하는 미디어 관리자(Media manager)[8], 그리고 데이터베이스와의 연동을 담당하는 데이터베이스 커넥터[9]이다. ISSA는 스트림의 제어를 위하여 IETF에서 제안한 RTSP를, 스트림과 세션의 표현과 초기화를 위해 SDP(Session Description Protocol)[10]와 SIP(Session Initialization Protocol)[11] 프로토콜을, 스트림 데이터의 실시간 전송을 위하여 RTP, TCP, UDP 프로토콜들을 지원한다. 또한, BeeHive와의 연동을 위하여 RPC (Remote Procedure Call)를 사용하는 BeeHive 커넥터를 개발하였다. 그리고 ISSA가 지원하는 미디어 포맷으로는 WAVE, MPEG-1/2/4, MP3 등이 있으며, VOD(Video on Demand), MOD(Music on Demand), 웹캐스팅(Webcasting)을 위한 개발자 라이브러리를 제공하고, 웹서버와 CORBA A/V스트리밍 서비스와의 연동을 위한 게이트웨이를 제공한다.

ISSA의 특징은 다음과 같다. 첫째, 플랫폼 독립적이다. ISSA는 Windows 및 Unix 기반의 운영체제 환경에서 작동할 수 있도록 설계되어 플랫폼에 독립적으로 작동한다. 둘째, 객체지향 멀티미디어 데이터베이스의 서비스를 지원한다. ISSA는 기존의 파일 시스템 및 관계형 데이터베이스 시스템 기반의 멀티미디어 통신 시스템의 한계를 벗어나 멀티미디어 데이터를 효율적으로 관리할 수 있는 객체지향형 멀티미디어 데이터베이스와인 BeeHive와의 연동 기능을 제공한다. 셋째, 유니캐스팅, 멀티캐스팅, 웹캐스팅의 지원한다. ISSA는 유니캐스팅 기반의 VoD 어플리케이션을 개발하기 위한 라이브러리를 제공하며, 인터넷 방송에서 사용하기 위한 멀티캐스팅 및 웹캐스팅 서비스 개발을 위한 라이브러리를 지원한다. 넷째, 멀티미디어 통신을 위한 표준 프로토콜의 지원한다. ISSA는 유니캐스팅 환경에서 VCR과 유사한 기능을 지원하기 위하여 IETF에서 제안한 표준프로토콜인 RTSP 프로토콜 라이브러리를 제공하며, 멀티미디어 데이터의 QoS를 보장하기 위하여 RTP 프로토콜 라이브러리를 제공한다. 또한, TCP 및 UDP를 통한 미디어 스트리밍을 지원하며 웹과의 연동을 위하여 HTTP 프로토콜을 지원한다. 다섯째, 다양한 표준 미디어 포맷의 지원한다. ISSA는 인터넷 및 인트라넷에서 고품질의 서비스를 제공하기 위하여 MPEG-1/2/4 등 표준 스트리밍 미디어 포맷을 지원하며, 음악 포맷으로 MP3를 지원한다. 그리고, 새로운 미디어 포맷을 쉽게 지원할 수 있도록 설계되어 있다.

본 논문의 구성은 다음과 같다. 2장에서는 멀티미디어 통신 시스템에 관한 관련연구들을 소개하고, 3장에서는 ISSA의 설계 모델에 대해서 설명한다. 4장에서는 ISSA의 각 구성요소의 구조와 기능에 대해서 설명하고, 5장에서는 ISSA를 이용한 멀티미디어 어플리케이션에 대해서 설명하고, 6장에서 결론을 맺는다.

Ⅱ. 관련 연구

본 장에서는 스트리밍 서비스와 관련된 멀티미디

어 통신 프레임워크에 대해서 살펴본다. CMT (Contineous media Toolkit)[12]는 미국 버클리 대학에서 만든 분산 멀티미디어 어플리케이션을 위한 개발 도구이다. CMT는 스크립트 언어이면서 GUI 도구인 Tcl/Tk와 RPC와 네이밍 서버가 포함된 네트워크 도구를 제공하는 Tcl-DP 위에서 구현되었다. CMPlayer는 MPEG-I video, MJPEG video, 8 KHz ulaw audio, 8 bit linear audio, 그리고 16 bit linear audio를 재생하는 어플리케이션으로 로컬 파일 시스템뿐만 아니라 원격의 비디오 파일서버로부터도 파일을 재생시킬 수 있다. 또한, Netscape 용 플러그인을 제작하여 웹과도 연동하고 있다. 그러나, CMT는 현재 데이터베이스와의 연동을 위한 기능은 제공하고 있지 않으며, 멀티캐스팅과 스트림의 제어를 위한 RTSP 프로토콜을 지원하고 있지 않다.

KISS(Communication Infrastructure for Streaming Services)[13]는 독일의 GMD에서 개발한 네트워크 서비스 응용의 투명한 통합을 지원하는 스트리밍 서비스를 위한 통신 인프라구조이다. 이는 네트워크의 조건과 사용자의 요구사항에 따라 실시간으로 컨텐츠를 스트리밍 한다. 컨텐츠 스트림은 요구된 QoS를 만족하기 위해 네트워크 경로 중에 몇 가지 변화가 있을 수 있기 때문에 네트워크 서비스와 연관되어 있다. 특히, 멀티캐스트인 경우에는 서로 다른 품질을 사용자들에게 전달할 수 있다. 서비스는 클라이언트와 서버 어플리케이션의 어떠한 개입도 없이, 그리고 품질을 결정하는 중앙의 구성요소 없이 동적으로 통합될 수 있다. KISS는 이기종의 멀티포인트 환경을 지원하고, 다중 사용자로의 확장을 지원한다. 그러나, KISS는 아직 PCM, MPEG 오디오 형식의 미디어만을 지원하도록 구현되었으며, NAP를 통한 성능의 오버헤드를 줄이는 것이 관건이다.

CORBA A/V Streaming Service[14]은 분산 객체 미들웨어인 CORBA[15]에서 오디오/비디오 데이터의 스트리밍을 제공하기 위한 서비스로서 OMG(Object Management Group)에서 제정한 스트리밍 서비스를 위한 모델이다. 이 모델은 둘 이상의 멀티미디어 디바이스 사이의 연속적인 미디어 전송을 스트림으로 정의하고 있다. 그리고, 스트림의 표현 방법, 스트림 제어를 위한 기본 오퍼레이션(시작/중지), 스트림 QoS의 표현과 변경, TCP/UDP/RTP 등 다양한 전송 프로토콜을 지원하기 위한 방법, 유니캐스트/멀티캐스트/브로드캐스트를 위한 오퍼레이션, 플로우의 동기화를 위한 오퍼레이션, 보안을 위한 암호키 설정 오퍼레이션 등을 정의하고 있다. 본 논문에서는 CORBA A/V 스트리밍 서비스와의 연동을 위한 ISSA/CORBA 게이트웨이의 구조를 제안한다. TAO[16]는 미국 워싱턴 대학에서 개발한 실시간 CORBA로서, TAO에서는 OMG에서 정의한 CORBA A/V 스트리밍 서비스를 구현하였는데, 이는 미국 오레곤 대학의 MPEG 플레이어[17]를 토대로 구현한 것이다.

Ⅲ. 통합 스트리밍 서비스 구조(ISSA)의 모델

ISSA 프레임워크 모델은 그림 1과 같이 크게 상위 계층의 스트리밍 응용, ISSA와 스트리밍 응용 사이의 인터페이스인 MOA(Media Object Architecture)[9], 데이터베이스 커넥터, 게이트웨이 모듈(Gateway Module), 그리고 ISSA로 구성되어 있다. MOA는 다시 응용 인터페이스(Application Interface)와 응용 Wrapper로 구성되어 진다. 응용 인터페이스는 AMS(Agent for Multimedia Service)와 스트리밍 서버(Streaming Server)로 나누어지는데, AMS의 역할은 각 스트리밍 서버에 분산되어 있는 컨텐츠의 정보나 세션의 정보를 통합, 분배하는데 있다. 스트리밍 서버의 역할은 모듈화 되어 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. Directshow 소스 필터는 Microsoft 사의 윈도우 Media Player와 연동 시에 사용되며, Winamp의 플러그인은 MP3를 서비스 할 때 사용된다. 또한, 데이터베이스 커넥터는 실시간 멀티미디어 데이터베이스인 BeeHive와 연동시켜주는 BeeHive 커넥터, 게이트웨이 모듈은 CORBA 연동을 위한 CORBA 게이트웨이로 구성된다.

그리고, 프레임워크의 핵심을 이루는 ISSA의 기본구조는 세션, 전송, 미디어 및 자원 관리자로 구성되어있다. 세션 관리자는 RTSP 기반의 멀티미디어 스트리밍 서비스의 세션 제어를 담당하며, 유니캐스팅과 멀티캐스팅을 지원할 수 있는 구조로 되어 있다. 또한, 데이터베이스의 트랜잭션 요청을 위해 RTTP, 컨텐츠 정보의 분배와 공유를 담당하는 SCP(Service Control Protocol)를 지원한다. 전송 관리자는 TCP 및 UDP, 그리고 RTP/UDP 프로토콜

을 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. 자원 관리자는 스트리밍 시스템에서 QoS의 명세화, 매핑, 모니터링, 제어 기능을 제공하며, 메모리 버퍼 관리, 쓰레드 스케줄링 등의 기능을 수행한다[7][8][9].

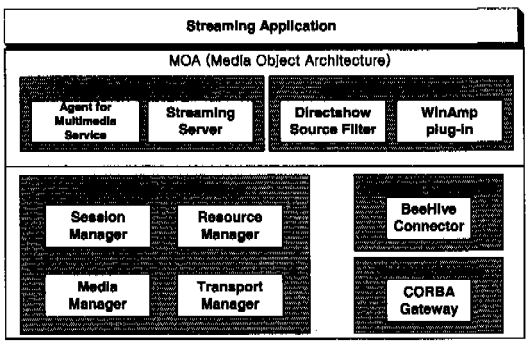


그림 1. 통합 스트리밍 서비스 프레임워크 구성

Ⅳ. ISSA의 구성 요소

본 장에서는 ISSA의 모듈 중에서 가장 핵심적인 세션관리자, 전송관리자, 미디어 관리자, BeeHive 케넥터, ISSA/CORBA 게이트웨이 등의 기능과 구조에 대해서 살펴본다.

4.1 세션 관리자(Session Manager)

ISSA의 세션관리자는 스트리밍 응용 프로그램 사이의 세션을 생성, 소멸하거나 재생, 멈춤 등의 기능을 제공함으로써 스트리밍 되는 미디어를 제어할 목적으로 주로 RTSP를 고려하여 설계되었다. 그러나, 실시간 데이터베이스인 BeeHive와의 연동 과정에서 데이터베이스 트랜잭션 처리를 위한 기능이 필요로 하게되었으며, 이러한 기능을 위해 RTTP 모듈이 세션 관리자에 추가되었다. RTTP는 원격지 클라이언트와 서버의 트랜잭션 요청과 결과 전달을 목적으로 하는데, 현재 Read, Write, Find, Play의 네 가지 트랜잭션을 처리할 수 있다. RTTP는 데이터베이스와 연동되어진 스트리밍 서버의 요청과 그에 대한 처리를 실시간으로 수행하기 위한 프로토콜이기 때문에 실시간 특성을 지닌 데이터 전달을 제어하기 위한 응용 레벨의 세션 프로토콜인 RTSP를 참조하여 구현되었다.

세션 관리자는 추가로 SCP를 지원하고 있으며, SCP는 멀티캐스트 채널 관리 기능과 다양한 미디어 정보를 효율적으로 분배 및 관리하고, 또한 스트리밍 서버를 관리할 목적으로 설계되었다. 그리고, 세션의 정보를 기술(description)하기 위한 표준 프로토콜인 SDP를 사용하여 세션을 기술함에 있어 표준화에 따른 범용성을 확보하였다. SDP는 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, 세션관리자는 세션 성립 이전과 성립 이후 관리에서도 편리성을 확보하기 위하여 여러 부가적인 프로토콜을 지원하고 있다. 이것은 세션 관리자의 성격이 응용 계층으로 확장된 것을 의미한다.

4.1.1 세션 관리자의 지원 프로토콜

세션 관리자는 초기에는 세션의 제어만을 목적으로 구현되었으나, 개발 과정에서 변화하는 환경을 수용하기 위해 여러 서비스 기능이 추가되었다. 이러한 서비스들은 업계에서 표준으로 사용하는 프로토콜을 우선하여 구현되었으나 그렇지 못한 경우에는 자체적으로 개발한 프로토콜을 사용하였다. 세션 관리자에서 사용된 표준 프로토콜은 대표적으로 RTSP, SDP등이 있으며 자체 제작한 RTTP 및 SCP도 사용한다.

4.1.1.1 RTSP와 SDP

RTSP는 오디오/비디오와 같이 시간적 동기화가 필요한 단수 혹은 복수 개의 연속된 미디어의 채널 설정과 제어를 위한 프로토콜로 멀티미디어 서버의 "Network Remote Control"과 같은 기능을 제공한다. 서버는 각 세션에 8자리의 식별자를 지정하여 관리하며, 연결지향형 (Connection-Oriented) 서비스나 비연결형(Connectionless) 서비스에 상관없이 동작한다. 그리고, 실제적인 미디어의 전송을 담당하는 전송 계층과는 독립적으로 작동한다.

그림 2에서는 RTSP를 이용한 스트리밍 과정을 보여준다. RTSP는 세션기술(Session Description)이라는 현재의 세션 정보를 먼저 얻어 오며, 이때는 HTTP나 여타 다른 방법을 사용할 수 있다. 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.

그리고 세션 기술에 사용되는 표준 프로토콜인 SDP는 RTSP와 마찬가지로 IETF의 RFC 문서에 의해 표준화 내용이 정의되어 있다. 이는 현재 스트리밍 세션 기술 프로토콜로 사용되고 있으며, 텍스트 기반의 프로토콜이라는 점과 스트리밍 전송 프로토콜과는 독립적이라는 점은 RTSP와 유사하다. 그렇지만, 프로토콜 자체의 전송에 있어서의 독립성을 가지고 있으며, 이때 사용되는 프로토콜로는 RTSP나 현재 표준화 진행중인 SAP(Session Announce Protocol) 프로토콜 등이 있다. ISSA에서는 RTSP와 SCP가 SDP의 전송에 사용되고 있다.

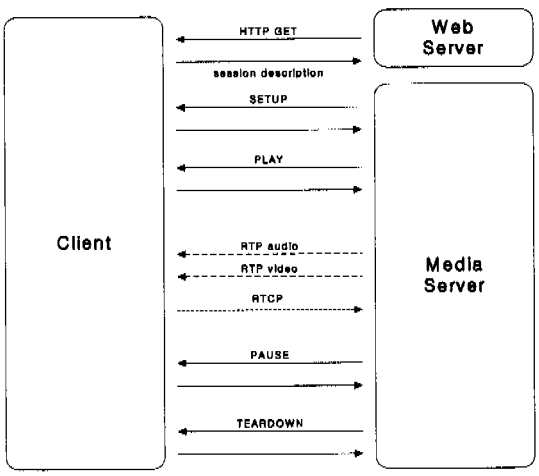


그림 2. RTSP를 사용한 스트리밍의 과정

4.1.1.2 RTTP

RTTP는 ISSA와 BeeHive와의 연동을 위해서 제안된 프로토콜이다. 일반적인 스트리밍 프레임워크에서는 스트리밍 데이터와 메타 데이터를 분리하고, 스트리밍 데이터 부분은 파일 시스템을 사용하여 저장하며, 메타 데이터 부분은 관계형 데이터베이스에 저장된다. 그리고, 파일 시스템과 관계형 데이터베이스를 동시에 사용하는 스트리밍 프레임워크 환경에서는 메타 데이터의 전송에 HTTP와 같은 프로토콜을 사용한다. 하지만, 멀티미디어 데이터베이스를 이용할 경우 스트리밍 데이터(대용량의 멀티미디어 자료)와 메타 데이터(소용량의 관련 정보 자료)는 같은 데이터베이스 내에 존재하며, 이러한 결과 메타 데이터는 스트리밍 데이터와 마찬가지로 실시간 전송을 보장할 수 있는 방법을 가지게 된다. HTTP를 사용하는 메타 데이터 전송 방법은 요청되거나 응답되는 패킷에 타임 스템프 기능이 없어 멀티미디어 데이터로부터 얻은 실시간 속성을 보장해 주기 어려운 문제가 있다. 따라서, ISSA와 같은 멀티미디어 데이터베이스를 지원해야하는 환경에서는 실시간적인 속성을 갖는 메타 데이터 전송 기능과 스트리밍 데이터의 트랜잭션 기능을 동시에 지원하기 위한 새로운 전송 프로토콜이 필요하게 되었다. RTTP는 이러한 목적 때문에 RTSP와 유사하게 구현되었고 트랜잭션 요청의 전송 역할만을 담당하며, 실제적인 트랜잭션은 BeeHive Connector에서 데이터베이스에 요청 시 이루어지게 된다.

4.1.1.3 SCP와 AMS

SCP는 ISSA의 응용 계층에 존재하는 AMS와 스트리밍 서버 응용간의 정보 공유 및 제어를 위해 설계된 프로토콜이다. AMS의 설계 목적은 그대로 SCP의 설계 목적으로 적용될 수 있는데, 이는 복수의 스트리밍 서버에서 컨텐츠와 컨텐츠 정보 공유, 그리고 현재 서비스되고 있는 채널(세션)에 대한 제어와 정보 제공을 목적으로 한다. 세션 관리자에서는 이러한 SCP의 메시지 형식을 정의하고 있으며, 이를 처리하기 위한 서버와 클라이언트 인터페이스를 제공해 준다. SCP에서의 메시지 설계 역시 RTSP, RTTP등과 같은 문법으로 설계되었나, 참여자에 대한 정보와 권한을 프로토콜 내에서 명시하고 있다는 점과, 오직 연결지향형 서비스만을 지원한다는 점이 다른 점이다. 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.

4.1.2 세션관리자의 구현

세션 관리자는 크게 인터페이스, 메시지, 세션 정보, 승인 제어 클래스 모듈로 나누어진다. 인터페이스 클래스 모듈은 세션 관리자 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 모듈은 인터페이스 클래스에서 사용되며 메시지를 생성, 해석하는 기능을 한다. 세션 클래스 모듈은 복수의 세션을 관리하며, 인터페이스 클래스 집합에 의해서 내용이 변경, 추가, 삭제된다. 그리고, 승인 제어 클래스 모듈은 스트리밍에 있어 최소한의 서비스품질 보장을 위한 승인 제어를 담당하고 있다.

4.2 전송 관리자(Transport Manager)

전송관리자는 다양한 형식의 멀티미디어 데이터를 전송하기 위해 적절하게 분해, 조립하는 기능과 패킷화된 미디어 데이터를 다양한 네트워크 환경에서 실시간으로 전송하는 역할을 수행한다. 즉, 전송관리자는 다양한 네트워크 환경에서 여러 종류의 멀티미디어 데이터를 스트리밍 하기 위한 유연하고 확장성 있는 전송 환경을 제공한다. 그림 3은 전송관리자의 전송 프로토콜 스택으로서 전송관리자는 RTP/UDP 및 TCP기반의 다양한 미디어 전송 프로토콜과 IP-유니캐스트 및 IP-멀티캐스트를 지원하며, ISSA의 세션관리자에서 제공하는 세션 제어 프로토콜에 관계없이 유연하게 사용할 수 있다.

RTP는 멀티미디어 데이터의 실시간 전송을 지원하는 프로토콜로서, 비디오와 같이 실시간 특성을 가진 멀티미디어 데이터의 종단간 실시간 전달 서비스를 제공한다. 이러한 실시간 전달 서비스는 TCP가 가지고 있는 오버헤드를 효율적으로 극복할 수 있으며, RTP 프로토콜은 페이로드 타입, 소스식별자, 시퀀스 번호, 타임 스탬프, 그리고 제어정보 모니터링을 위한 RTCP 등을 이용하여 실시간 전송 기능을 지원한다. 일반적으로 RTP는 UDP 위에서 동작하나, ATM 등과 같은 기타 다른 전송 환경 위에서도 동작할 수 있게 설계되었다. 또한 RTP를 이용하면 MBone (Multicast Backbone)과 같이 IP-멀티캐스트를 지원하는 네트워크에서 멀티캐스트 전송 기능을 사용하여 하나의 패킷을 특정 그룹의 다중 사용자에게 한번에 모두 전송하는 것이 가능하다. 하지만 RTP 프로토콜 자체로는 QoS를 보장할 수 없으며, 다만 다른 제어 기법 등을 사용하여 QoS를 보장하는 것은 가능하다. 한편 RTCP 프로토콜은 하나의 RTP 세션과 같이 사용되며, QoS 정보를 모니터링하고 진행중인 세션에 참가한 사용자들의 정보를 전달하는 것을 목적으로 한다. 즉 RTP는 전송 기능만을 가지며 RTCP는 전송세션에 대한 모니터링 기능을 수행한다.

4.2.1 네트워크 인터페이스

ISSA는 다양한 네트워크 환경에 대하여 투명성을 제공하기 위해 그림 4와 같이 네트워크 인터페이스를 통해 네트워크 송수신 기능을 수행한다. 네트워크 인터페이스는 네트워크 환경에 독립적으로 실행될 수 있고 다양한 네트워크 환경을 지원하는 일종의 네트워크 wrapper 인터페이스로서, 현재 윈도우 환경의 WinSock 및 유닉스 환경의 버클리 소켓에 대한 인터페이스를 제공하며, 향후 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 손쉽게 구현할 수 있으며, ISSA 프레임워크의 전송관리자나 세션관리자와 같이 네트워크 사용이 많은 다른 모듈에서 이용되어 프로토콜의 독립성을 지원 해줄 뿐 만 아니라 소스코드 레벨에서 이기종간 플랫폼의 호환성도 보장해 준다.

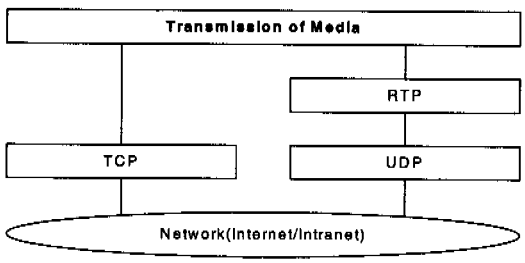


그림 3. 전송관리자의 전송프로토콜 스택

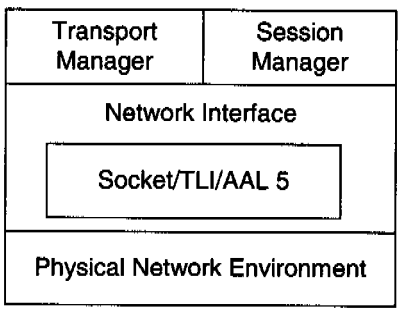


그림 4. 네트워크 투명서 제공을 위한 네트워크 인터페이스

4.2.2 전송관리자 구조

전송관리자는 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.

4.2.3 전송관리자 구현

전송관리자의 구현은 윈도우와 유닉스 환경에서 모두 동작할 수 있도록 ANSI C++ 언어로 구현하였으며, 플랫폼 의존적인 코드는 다양한 플랫폼으로

이식 가능한 쓰레드, 타이머, 네트워크 인터페이스 라이브러리를 만들어 이를 사용함으로써 쉽게 구현이 가능하게 하였다. 네트워크 인터페이스는 윈도우 환경의 Winsock과 유닉스 환경의 berkeley 소켓을 지원하여 하나의 일관된 인터페이스로 네트워크에 대한 접근할 수 있도록 해준다. 또한 타이머의 경우 패킷을 주기적으로 전송하기 위해 필요한 기법으로서, 미디어 데이터 패킷이나 혹은 제어 패킷이 서버 측으로부터 클라이언트 측으로 Push 방식을 이용하여 전송될 때 사용된다.

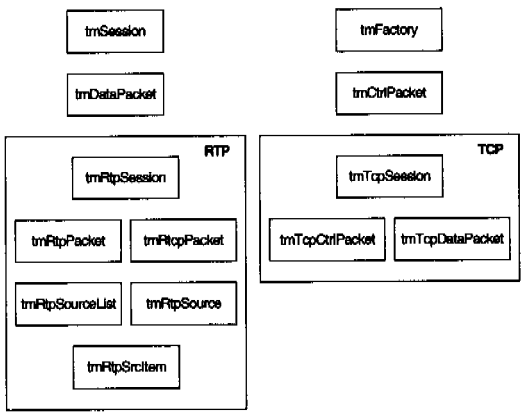


그림 5. 전송관리자의 전체 모듈 구성도

그림 5는 구현된 전송관리자의 전체 모듈 구성도로서 총 13개의 모듈로 구성되며, RTP와 관련 있는 6개의 모듈과 TCP와 관련된 3개의 모듈, 그리고 상위 레벨의 추상적인 모듈 4개로 구분된다. 전송 프로토콜을 가동시키는 프로그래밍 인터페이스는 앞장에서 설계된 대로 매우 단순하고 유연하게 구현되었으며, 데이터를 보내는 서버 프로그램의 경우 InitSender()로 세션을 초기화시키고, 클라이언트의 경우는 InitReceiver()로 초기화시키며, 그 후 Run()을 통해 세션을 동작시킬 수 있다. 이 때 서버는 미디어 데이터를 얻어올 미디어 소스, 클라이언트는 전달된 데이터를 표현할 미디어 싱크의 객체를 넘겨받는다. 전송 세션이 동작중일 때 Pause()를 통해 잠시 중단시키거나 Deinit()를 통해 완전히 정지시킬 수 있다. 따라서 전송관리자는 응용 프로그램에서 사용할 전송 프로토콜에 따라 해당 세션을 생성하고 위와 같은 인터페이스를 통해 전송을 쉽게 제어할 수 있도록 구현되었다.

4.3 미디어 관리자(Media Manager)

ISSA에서 미디어 관리자는 파일, 멀티미디어 데이터베이스, 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.

미디어 관리자는 그림 6에서 보는 바와 같이 크게 미디어 소스와 미디어 싱크의 두 가지 모듈로 구성되어 있다. 미디어 소스 모듈의 가장 중요한 기능은 입력된 미디어를 추상화시키는 것으로써, 멀티미디어 데이터베이스인 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 데이터를 가장 적당한 디코딩 방법을 선택할 수 있는 분배기 역할을 한다. 그리고, 클라이언트 측에서 전송 관리자로부터 수신된 미디어 데이터를 넘겨받아 적절한 오디오/비디오 장치로 재생하는 역할 또한 가지고 있다.

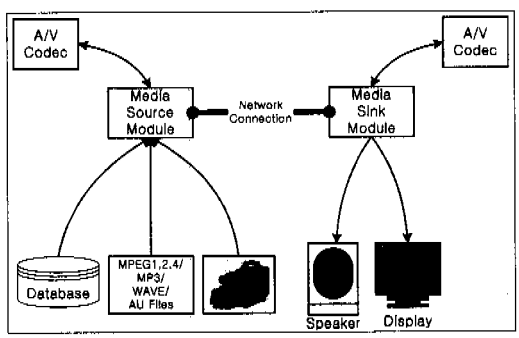


그림 6. 미디어 관리자의 구조

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.

제안된 미디어 관리자는 라이브 비디오 카메라와 마이크를 사용하는 실시간 화상회의 미디어 스트림,

파일 시스템의 미디어 파일, 멀티미디어 데이터베이스의 관리를 받는 미디어까지 처리할 수 있을 뿐만 아니라, 향후 새로운 형태의 미디어 타입이 출현하더라도 이를 쉽게 수용할 수 있는 유연성과 확장성을 가지게 된다.

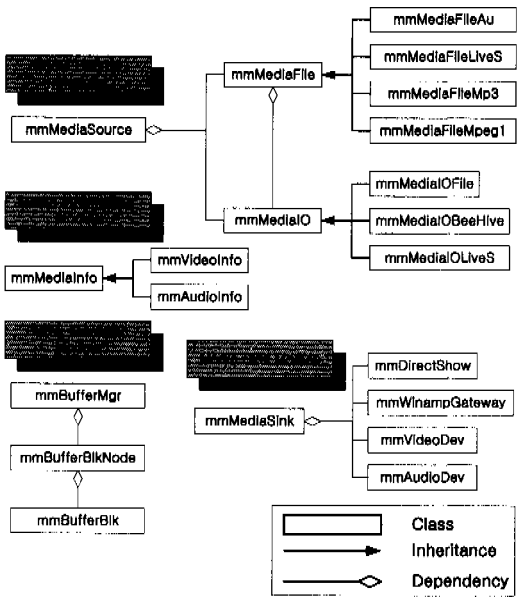


그림 7. 미디어 관리자의 클래스 다이어 그램

미디어 관리자의 클래스 다이어그램은 그림 7에서 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, 데이터베이스, 마이크와 같은 다양한 미디어 데이터를 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.

4.3.1 소스필터

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 8과 같으며 미디어를 읽어들이는 소스 필터와 미디어 데이터를 알맞은 소프트웨어 디코더와 연결하여 미디어를 디코딩하여 디바이스에서 디스플레이하기에 적당한 형태로 변환해주는 트랜스폼 필터, 그리고 변환된 데이터를 디바이스로 출력하는 랜더러 필터로 구성된다[8].

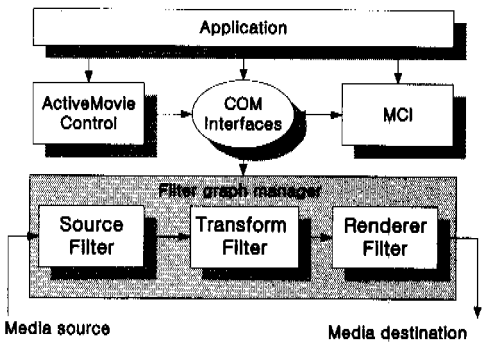


그림 8. DirectShow의 구조

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, 바로비전사에서 개발된 MPEG-2 소프트웨어 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 DirectShow의 필터구조를 지원하는 API를 제공한다면 이것 역시 개발된 RTP 소스 필터와 함께 이용될 수 있다. 또 다른 응용으로는 바로비전사에서 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 확

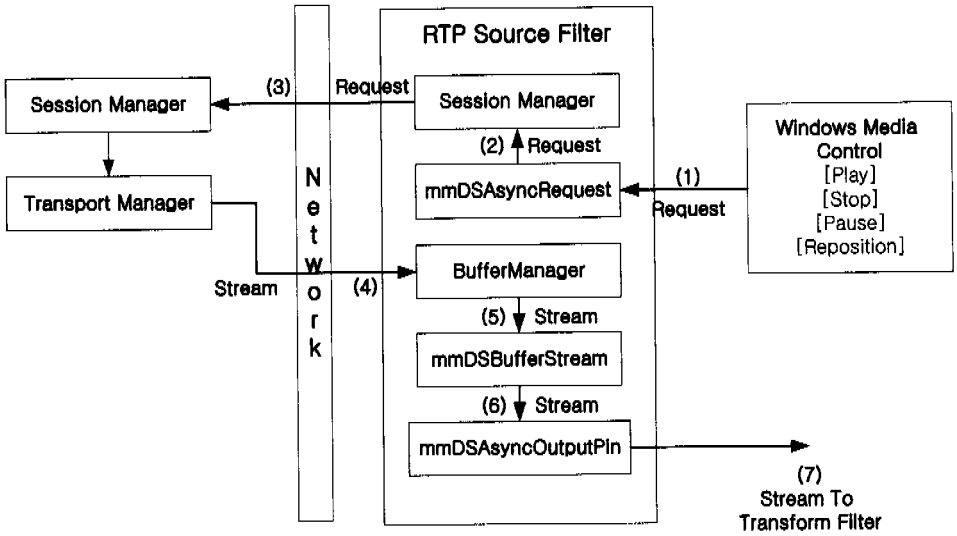


그림 9. RTP 소스 필터의 내부구조 및 동작 순서

장성과 응용성을 가지는 이유이다.

소스필터의 역할은 로컬 디스크나 네트워크 또는 인터넷으로부터 데이터를 가져와 트랜스포트필터로 전달 역할을 수행하는 DirectShow의 COM 객체이다. 그림 9는 ISSA 프레임워크에서 개발된 RTP 소스 필터의 내부구조 및 작동 순서를 보여주는데, 작동의 순서는 각 객체를 이어주는 화살표에 붙어있는 번호로 보여준다.

Windows Media Control 객체에서 PLAY, STOP, PAUSE, REPOSITION등과 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 전달하게된다. 이렇게 전달되어진 미디어 스트림은 소스필터의 출구인 mmDSAsyncOutputPin을 통해서 다음에 단계에 있는 DirectShow의 트랜스폼 필터에게 전달한다. 트랜스폼 필터에 의해 디코딩 되어진 미디어 데이터는 디스플레이 디바이스와 오디오 디바이스를 통하여 재생되어진다. 즉 미디어 타입을 선택한 소스필터에 의해 미디에 타입에 알맞게 필터 그래프가 구성되고 그 필터그래프를 통하여 재생이 되어지는 것이다.

현재 RTP 소스 필터에서 미디어 스트림에 관련되어 사용자가 선택할 수 있는 요구사항은 PLAY, STOP, PAUSE의 3가지 종류이다. PLAY는 미디어 재생을 위한 요구이고, STOP은 미디어 재생을 끝마치고 미디어 전송에 관련된 정보를 가진 서버 세션이 더 이상 필요하지 않을 때의 요구이며, PAUSE는 미디어 전송에 관련된 정보를 가진 서버 세션과의 연결은 유지한 채 잠시 미디어 재생을 멈춘 상태이다. PAUSE 다음 요구가 PLAY인 경우에는 멈춘 지점부터 다시 재생이 되며, 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 보내게 된다. 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 소스 필터의 Output 핀을 지나서 디코딩을 위한 트랜스폼 필터로 이동하게 된다.

4.3.2 버퍼 관리자

ISSA는 클라이언트측의 mmMediaSink에서 동작하며 서버로부터 전송되는 미디어 스트림 데이터를

버퍼링 하고, 버퍼에 있는 데이터를 Push 및 Pull 방식으로 미디어 재생 장치로 전달하는 기능을 지원하는 버퍼관리자를 제공한다. 버퍼관리자는 클라이언트에서의 패킷 수신 버퍼가 Push 방식과 Pull 방식의 데이터 출력을 단일한 인터페이스로 제공하여 다양한 미디어 재생 장치를 쉽게 지원할 수 있고, 버퍼링 기법을 제공하여 종단간 지터를 적응시킬 수 있다. 제안된 버퍼링 기법은 다양한 미디어 장치마다 그에 알맞는 버퍼 passing을 위하여 stub와 같은 변환기를 설치해야하는 비효율적인 중복성을 피할 수 있었으며, 더 나아가 네트워크 지터를 적응하기 위한 버퍼링 기능도 같이 제공함으로써 버퍼링을 따로 수행하는 것에 비해 메모리 사용의 효율을 높일 수 있다. 클라이언트에서의 Push/Pull 버퍼 관리 기법의 전체적인 동작 구조는 그림 10과 같이 이루어지며, 실제 클라이언트에 위치한 Push/Pull 버퍼 관리 기법이 미디어 재생 장치와 연관되어 동작하는 구조를 표현한 것이다.

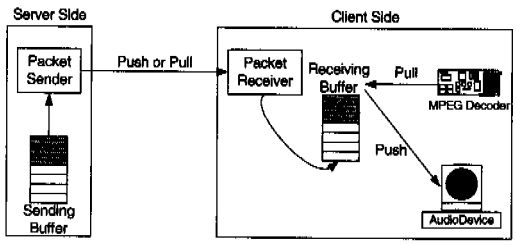


그림 10. Push/Pull 버퍼 관리 기버의 동작 구조

서버 측의 패킷 송신자를 통해 Push나 Pull 방식으로 전송된 패킷은 클라이언트에서의 패킷 수신자를 통해 수신되어 해석된 후 수신 버퍼로 전달되며, 수신 버퍼로 전달된 미디어 스트림 데이터는 설정된 미디어 재생 장치로 초기에 설정된 Push 혹은 Pull의 데이터 전달 방식을 이용해 재생된다. 즉 버퍼 관리 기법은 초기에 교섭을 통해 미디어 표현장치에 알맞은 데이터 전달 방식을 설정한 후 전송 세션이 시작하면 이미 설정된 전달 방식에 따라 데이터를 적절한 장치로 전달하며, 이 때 미디어 재생 장치는 설정된 데이터 전달 방식을 지원할 수 있는 장치이다. 또한 이때 버퍼로 데이터를 입력해주는 Packet Receiver는 서버로부터 데이터를 Push 방식으로 받을 수도 있으며 Pull 데이터 요청을 처리하는 stub을 설치하면 Pull 방식으로도 데이터를 받을 수 있어, 실제 수신 버퍼는 입력 데이터의 네트워크 전달 방식과 상관없이 유연하게 동작한다.

4.4 BeeHive 커넥터

이 절에서는 ISSA와 객체지향형 실시간 멀티미디어 데이터베이스인 BeeHive를 연동하는 모듈로서 BeeHive 커넥터를 소개한다. 스트리밍 시스템과 멀티미디어 데이터베이스를 연동하는 경우 스트리밍 중에도 재생중인 미디어에 관련된 정보들을 데이터베이스로부터 검색이 가능하여 다양한 멀티미디어 데이터베이스 서비스를 제공할 수 있다는 잇점을 가지게 된다. 메타데이터를 데이터 베이스에 저장하고 스트리밍 데이터 자체는 파일로 제공하는 관계형 데이터베이스와 연동방식과는 달리 제안한 연동 기법에서는 BeeHive에서 처리 가능한 트랜잭션을 정의하고, 트랜잭션 처리를 위한 RPC(Remote Procedure Call) 인터페이스를 정의하였다. 연동 구현을 위한 구체적인 응용 분야로 영화 데이터베이스를 선택하였으며, BeeHive에서 영화 객체(Movie Object)를 저장하고 다룰 수 있는 읽기(Read), 쓰기(Write), 찾기(Find) 그리고 재생(Play) 트랜잭션을 정의하였다.

ISSA와 BeeHive가 상호 연동되어 작동할 때의 모델은 그림 11과 같이 트랜잭션 인터페이스와 스트리밍 인터페이스로 나눌수 있다. 트랜잭션 인터페이스는 BeeHive 트랜잭션을 ISSA 클라이언트에서 요청하고 이를 처리하기 위한 일련의 과정을 다루는 인터페이스이며, 사용자로부터 트랜잭션 요청을 받는 GUI, ISSA 클라이언트와 서버 사이의 트랜잭션 처리를 위한 RTTP, ISSA 서버와 BeeHive 사이의 RPC로 구성된다. 스트리밍 인터페이스는 ISSA 클라이언트와 서버 사이의 미디어 스트림을 처리하기 위한 인터페이스를 말하는데 ISSA 클라이언트의 미디어 출력을 위한 인터페이스와 ISSA 서버와 클라이언트 사이의 RTSP, RTP 세션, ISSA 서버와 BeeHive 사이의 스트리밍 전송 인터페이스로 구성된다. ISSA 서버의 컨텐트 관리자의BeeHive 캐넥터는 BeeHive와의 연동을 처리한다.

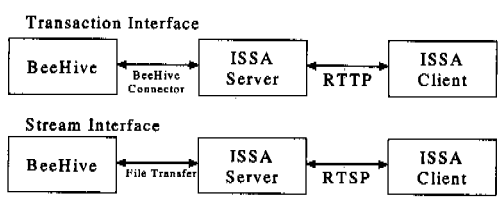


그림 11. ISSA and BeeHive 사이의 트랜잭션과 스트리밍 인터페이스

BeeHive 커넥터는 RTTP 프로토콜로 전달된 트랜잭션을 실제 해당되는 BeeHive 트랜잭션을 요구하게 된다. RTTP는 ISSA와 실시간 멀티미디어 데이터베이스인 BeeHive와의 연동을 위해 제안된 프로토콜로 데이터베이스의 트랜잭션 처리가 용이하도록 전달과 응답을 고려해 설계되었다. RTTP는 원격지 클라이언트와 서버의 트랜잭션 요청과 결과 전달을 목적으로 하는데, 현재 읽기, 쓰기, 찾기, 재생의 네 가지 트랜잭션을 처리할 수 있다. RTTP는 데이터베이스와 연동되어진 스트리밍 서버의 요청과 그에 대한 처리를 실시간으로 수행하기 위한 프로토콜이기 때문에 실시간 특성을 지닌 데이터 전달을 제어하기 위한 응용 레벨의 세션 프로토콜인 RTSP를 참조하여 구현되었다. RTTP는 ISSA 환경에서 BeeHive의 트랜잭션을 요청하고 처리하는데 사용하는 것으로 BeeHive와는 직접적으로 연관되지 않는다.

일반적인 스트리밍 프레임워크에서는 스트리밍 데이터와 메타 데이터를 분리하고, 스트리밍 데이터 부분은 파일 시스템을 사용하여 저장하며, 메타 데이터 부분은 관계형 데이터베이스에 저장된다. 하지만, 멀티미디어 데이터베이스를 이용할 경우 스트리밍 데이터(대용량의 멀티미디어 자료)와 메타 데이터(소용량의 관련 정보 자료)는 같은 데이터베이스 내에 존재하게 된다. 반면에, 파일 시스템과 관계형 데이터베이스를 동시에 사용하는 스트리밍 프레임워크에서 메타 데이터의 전송은 HTTP와 같은 프로토콜을 사용한다. 그러나, HTTP를 사용하는 메타데이터의 경우 요청되는 패킷에 타임 스탬프 기능이 없어 실시간 속성을 보장해 주기 어려운 문제가 있었다. 따라서, ISSA와 같은 멀티미디어 데이터베이스를 지원해야하는 환경에서는 실시간적인 속성을 갖는 메타 데이터 전송 기능과 스트리밍 데이터의 트랜잭션 기능을 동시에 지원하기 위한 새로운 전송 프로토콜이 필요하게 되었다.

미디어 스트리밍을 위한 인터페이스는 3단계를 거쳐 성능이 개선되었다. 첫 번째 단계는 파일로 저장한 뒤 파일명을 전송하는 것으로 매우 원시적인 미디어 전송 메커니즘이다. 두 번째 단계는 RPC 메소드를 통하여 스트리밍 데이터를 전송 받는 단계로 이를 위하여 BeeHive 내부에서 미디어 데이터를 규격화된 크기의 데이터로 나누어 저장하고 이를 RPC 메소드를 통하여 요구하면 블록 단위로 데이터를 요구하고 전송하는 방식이다. 세번째 단계는 미디어 데이터 블록의 BeeHive ID 리스트를 전달받아서 스트리밍 데이터를 직접 BeeHive로부터 받아오는 단계이다.

그러나, BeeHive 커넥터가 데이터 전송 시 RPC를 사용하는 경우 다중의 사용자를 지원하지 못하고, BeeHive와 ISSA의 코드가 서로 독립되지 못하다는 제약점을 가지고 있다. 이를 개선하기 위하여 ISSA와 BeeHive 서버 프로세스들이 모두 동일한 호스트에 존재한다고 가정하고 IPC(Inter Process Communication)와 더블버퍼링 기법을 사용하는 연동 기법 고려할 수 있다.

4.5 ISSA/CORBA 게이트웨이

분산객체 미들웨어의 표준인 CORBA 표준을 정의하는 OMG(Object management Group)에서는 멀티미디어 데이터의 스트리밍을 위해 CORBA A/V 스트리밍 서비스를 정의하였는데, 이는 멀티미디어 스트림 제어를 위한 기본 인터페이스를 정의함으로써 서로 다른 환경에서 작동하는 스트리밍 시스템 간의 상호운용성을 제공한다.

본 논문에서는 CORBA 서버인 동시에 ISSA 클라이언트로 동작하면서 A/V 스트리밍 서비스를 지원하는 ISSA/CORBA 게이트웨이를 제안한다. 제안된 ISSA/CORBA 게이트웨이는 A/V 스트리밍 서비스와 CORBA 상에서의 데이터베이스 트랜잭션 기능을 지원한다. 이는, CORBA의 IIOP(Internet Inter-ORB Protocol)를 통해 수신한 제어 메시지를 ISSA에서 사용하는 프로토콜인 RTSP와 RTTP에서 사용하는 적절한 메시지로 변환하는 역할을 하게 된다.

ISSA/CORBA 게이트웨이를 사용하면 CORBA 환경에서 동일한 인터페이스를 통해 서로 다른 환경의 클라이언트에서 ISSA 스트리밍 서버에 연결할 수 있는 방법을 제공해 준다. 또한, 스트리밍의 제어 인터페이스를 통해서 전송 프로토콜을 명시할 수 있으며, 멀티미디어 데이터의 전송은 IIOP를 사용하지 않고, RTP와 같은 멀티미디어 스트리밍에 적합한 프로토콜을 사용할 수 있게 함으로써 보다 나은 스트리밍 서비스를 제공할 수 있다.

CORBA A/V 스트리밍 서비스를 사용하여 스트리밍 어플리케이션을 개발할 때 스트림의 설정과 제어, 관리는 CORBA IIOP 프로토콜을 이용한다. 그러나, CORBA는 대단위 멀티미디어 데이터의 전송에 적합하지 않다. 따라서, 스트림의 전송은 RTP 등 멀티미디어 전송에 적합한 프로토콜을 사용하게 된다. ISSA/CORBA 게이트웨이는 CORBA 서버의

역할을 하는 동시에 ISSA 클라이언트 역할을 담당한다. 즉, CORBA의 RPC(Remote Procedure Call)을 ISSA 프레임워크에 맞는 메시지로 변환하는 과정을 처리한다. 게이트웨이를 사용함으로써 기존의 ISSA 프레임워크를 수정하지 않고 바로 분산 객체 미들웨어의 표준인 CORBA와 연동할 수 있다. 게이트웨이를 사용함으로써 요구에서 응답까지 걸리는 시간이 지연될 수 있으나, 이는 제어 메시지에 국한되기 때문에 오버헤드는 그리 크지 않다.

ISSA/CORBA 게이트웨이의 구조는 그림 12, 그림 13과 같이 표현할 수 있다. ISSA/CORBA 게이트웨이는 CORBA A/V 스트리밍 서비스를 지원하는 CORBA 서버인 동시에 RTSP, RTTP 등을 사용하는 ISSA 클라이언트 역할을 하게 된다. 클라이언트인 미디어 플레이어는 ISSA 라이브러리를 통해 구현될 수도 있고, Windows Media Player 등 기존의 미디어 플레이어일 수도 있다. 이때, 클라이언트는 CORBA 환경에서 작동하여야 한다. CORBA A/V 스트리밍 서비스를 사용하는 클라이언트에서 스트림을 요청했을 때, 게이트웨이는 클라이언트로부터 받은 CORBA의 요구 메시지를 ISSA에서 사용하는 요구 메시지 포맷으로 변경하여 ISSA 서버에 전달한다. 또한, ISSA 서버로부터 수신한 메시지를 CORBA 메시지로 변경하는 기능을 제공한다.

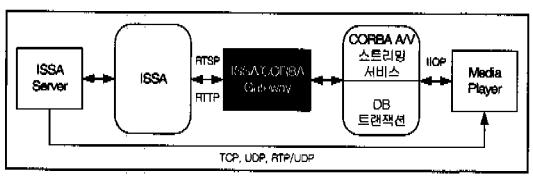


그림 12. ISSA/CORBA 게이트웨이의 구조

ISSA는 스트리밍 서비스뿐만 아니라 데이터베이스 트랜잭션 처리 기능도 제공해 준다. 그러나, CORBA A/V 스트리밍 서비스는 이를 지원하지 않기 때문에 클라이언트는 데이터베이스 트랜잭션을 위한 CORBA 인터페이스를 제공해줘야 하며 게이트웨이도 이를 지원할 수 있어야 한다. 이를 위해 CORBA IDL 언어를 이용하여 RTTP 인터페이스를 구현하였다.

RTP는 멀티미디어 데이터의 실시간 전송을 지원하는 프로토콜로서, 비디오와 같이 실시간 특성을 가진 멀티미디어 데이터의 종단간 실시간 전달 서비스를 제공한다. 이러한 실시간 전달 서비스는 TCP가 가지고 있는 오버헤드를 효율적으로 극복할

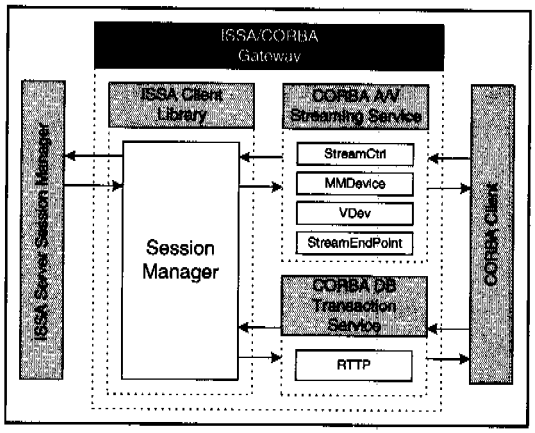


그림 13. ISSA/CORBA 게이트웨이와 클라이언트 사이의 구조

수 있으며, RTP 프로토콜은 페이로드 타입, 소스식별자, 시퀀스 번호, 타임 스탬프, 그리고 제어정보 모니터링을 위한 RTCP 등을 이용하여 실시간 전송 기능을 지원한다. 일반적으로 RTP는 UDP 위에서 동작하나, ATM 등과 같은 기타 다른 전송 환경 위에서도 동작할 수 있게 설계되었다. 또한 RTP를 이용하면 MBone (Multicast Backbone)과 같이 IP-멀티캐스트를 지원하는 네트워크에서 멀티캐스트 전송 기능을 사용하여 하나의 패킷을 특정 그룹의 다중 사용자에게 한번에 모두 전송하는 것이 가능하다. 하지만 RTP 프로토콜 자체로는 QoS를 보장할 수 없으며, 다만 다른 제어 기법 등을 사용하여 QoS를 보장하는 것은 가능하다.

V. ISSA를 이용한 멀티미디어 어플리케이션

본 논문에서 제안한 세션 관리자와 전송 관리자의 구현 사례를 보여 주기 위해서 그림 14와 같은 스트리밍 서비스 환경을 구축하였다. 오디오 및 비디오의 다양한 스트리밍 서비스를 위해서 서버는 AMS와 스트리밍 서버를 ISSA 프레임워크를 사용하여 구축하였으며, 클라이언트의 어플리케이션 경우 윈도우 Media Player를 위한 소스 필터와 WinAmp를 위한 플러그 인을 개발하여 사용하였다. 그리고, 사용자에게 스트리밍 서버와 컨텐츠의 투명성을 보장하기 위해서 웹 서버를 사용하였다.

그림 14는 ISSA기반의 클라이언트/서버 구조를 나타낸 것으로 AMS, 웹 환경과의 연동 과정과 정보, 제어, 스트리밍 흐름(flow)을 보여주고 있다. 정보 흐름은 원으로 표현되어 있고, 원 안의 숫자는

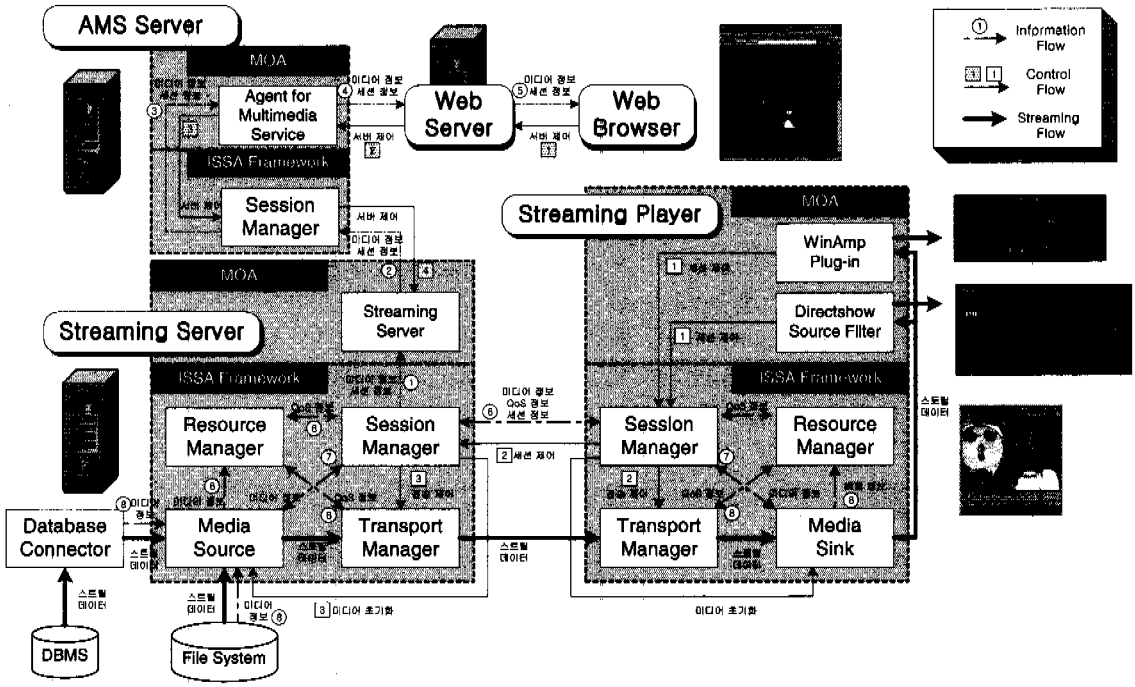


그림 14. 스트리밍 서비스의 작동 시나리오

흐름의 순서를 보여 준다. 정보 흐름은 스트리밍 서버가 가동되었을 때 자신이 가진 미디어 정보를 AMS 서버에 등록하는 것과, 등록된 미디어 정보가 웹에 의해 사용자에 알려주는 것을 보여 준다. 이때 사용자는 세션 관리자를 통해 원하는 컨텐츠의 미디어 정보를 이용하여 스트리밍 서버와 세션을 설정한다. 제어 흐름은 네모로 나타나져 있는데, 음영으로 표시된 네모는 웹을 통한 서버의 제어를 나타내며, 그렇지 않은 네모는 스트리밍 플레이어를 통한 세션의 제어를 나타낸다. 스트리밍 흐름은 굵은 선으로 표시되어 있는데, 데이터베이스나 파일 시스템에 존재하는 컨텐츠가 스트리밍 서버를 통하여 클라이언트 측에 스트리밍 되는 경로를 보여주고 있다.

정보 흐름은 스트리밍 서버가 가동되었을 때 시작되며, 이때 스트리밍 서버 세션 관리자는 스트리밍 서버 응용으로 컨텐츠의 정보를 보내주게 되며, 스트리밍 서버 응용은 이것을 AMS에 등록하게 된다. 음영 네모로 표기된 제어 흐름은 사용자가 웹을 통해 AMS에 스트리밍 서버의 제어를 명령하면 AMS는 이를 서버에 전달 한 뒤 제어 내용을 반영하게 된다. 그리고 흰 네모로 표기된 제어 흐름은 스트리밍 플레이어를 이용하여 사용자가 세션을 제어하고자 할 때 시작된다. 이때, 스트리밍 플레이어의 세션 관리자와 스트리밍 서버의 세션 관리자는 RTSP를 사용하여 세션을 제어하게 된다. 마지막으로 스트리밍 흐름은 하나의 세션이 성립된 후, 미디어 소스를 통해 데이터베이스나 파일 시스템에 저장되어 있는 컨텐츠가 스트리밍 되기 시작한다.

사용자는 홈페이지나 윈도우 Media Player를 직접 사용하여 서버에 스트리밍 요청을 보낼 수 있으며, 직접 서버에 요청을 할 경우 RTSP의 URI를 입력하면 서비스가 가능하다. 또한 사용자는 웹을 통해 현재의 서비스 상황이나 컨텐츠의 정보를 쉽게 접할 수 있다. 이것은 세션 관리자의 확장 역할로서 AMS 서버, 웹 서버가 스트리밍 서버의 위치와 컨텐츠를 부명화 시키고 있음을 뜻한다. 그리고, 이러한 서비스는 멀티캐스트 서비스나 유니캐스트 서비스에 상관없는 유연한 서비스 구조이다.

VI. 결 론

본 논문에서는 이기종 환경을 통합하고 다른 시스템과의 연동이 용이한 통합 스트리밍 서비스 구조의 기능과 설계 방법에 대해서 기술하였고, 이를 이용한 개발 사례를 설명하였다. 제안된 통합 스트

리밍 서비스 구조는 내부 구조를 객체 지향적인 모듈 구조로 설계함으로써 유연성, 확장성을 제공하며, 다양한 미디어 형식과 이기종의 네트워크 환경을 지원한다. 또한 상위 레벨의 응용 프로그램에 대한 고수준의 인터페이스인 MOA를 제공함으로써, 스트리밍 응용 프로그램을 위한 쉬운 개발 환경을 제공해준다.

ISSA는 다른 스트리밍 시스템에서 제공하지 못했던 객체지향 실시간 멀티미디어 데이터베이스인 BeeHive와의 연동 기능을 제공함으로써, 대용량 멀티미디어 데이터를 쉽고 효율적으로 관리할 수 있는 메커니즘을 제공하고 있다. 그리고, 클라이언트 부분은 범용성을 제공하기 위해 많이 사용되고 있는 Microsoft 사의 Windows Media Player를 지원하고자 RTSP 및 RTP 소스 필터를 개발하였으며, Winamp와의 연동 기능도 제공하고 있다.

향후 연구로는 자원관리자의 기능을 강화하고, 디지털 미디어의 저작권 및 전자상거래 상에서의 사용자 비밀 유지를 위한 보안 스트리밍 솔루션을 개발하는 것이다. 이를 위해 기존의 ISSA에 QoS 제어 및 관리기능, 인증, 암호화, 접근 제어 등의 기능을 추가한 모듈들을 구현할 예정이다.

참 고 문 헌

[1] RealNetworks, RealSystems G2, http://www.realnetworks.com

[2] Microsoft, Windows Media Technologies, http://www.microsoft.com/windowsmedia/.

[3] Apple, QuickTime Streaming Server, http://www.apple.com/quicktime/servers/.

[4] H. Schulzrinne, A. Rao, R. Lanphier, Real-Time Streaming Protocol(RTSP), IETF RFC 2326, April 1998.

[5] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, IETF RFC 1889, January 1996.

[6] J. Stankovic, S. Son and J. Liebeherr, "BeeHive: Global Multimedia Database Support for Dependable, Real-Time Applications", In Proc. of Second Workshop an Active Real-Time Databases, Lake Como, Italy, September 1997.

[7] 임익진, 정찬균, 이승룡, "멀티미디어 스트리밍 프레임워크에서 전송 및 세션 관리자의 설계 및 구현", 정보과학회 논문지 심사중.

[8] 이재욱, 이승룡, 홍인기, "스트리밍 프레임워크에서 미디어 관리자의 설계 및 구현", 정보과학회 논문지 심사중

[9] 홍영래, 김형일, 이승룡, "멀티미디어 스트리밍 프레임워크에서 컨텐츠 관리자의 설계 및 구현", 정보처리학회 멀티미디어 특집 논문집, 2000년 3월.

[10] M. Handley, V. Jacobson, "SDP: Session Description Protocol", IETF RFC 2327, April 1998.

[11] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg, "SIP: Session Initiation Protocol", IETF RFC 2543, March 1999.

[12] Ketan Mayer-Patel and Larry Rowe, "Design and Performance of the Berkeley Continuous Media Toolkit", Multimedia Computing and Networking 1997, Proc. SPIE 3020, pp 194-206.

[13] 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.

[14] S. Mungee, N. Surendran, and D. C. Schmidt, "The Design and Performance of a CORBA Audio/Video Streaming Service", In Proc. of the 32st Hawaii International Conference on System Systems(HICSS), Hawaii, Januarry, 1999.

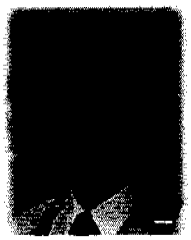
[15] Object Management Group, Control and Management of A/V Streams specification, OMG Document telecom/97-05-07 ed., October 1997.

[16] Douglas C. Schmidt, David L. Levine, Sumedh Mungee, Rajeev Bector, Chris Cleeland, and Irfan Pyarali, "Architectures and Patterns for High-performance, Real-time CORBA Object Request Brokers", Computer Communications Journal, 1998.

[17] Shanwei Cen, Calton Pu, Richard Staehli, Crispin Cowan and Jonathan Walpole, "A Distributed Real-Time MPEG Video Audio Player", Proceedings of NOSSDAV'95. April

18-21, 1995.
[18] Microsoft Corps., Introduction to DirectShow, http://www.microsoft.com/directx/dxm/help/ds/.

이 승 룡(Sungyoung Lee) 정회원



1978년 : 고려대학교 재료공학과 졸업
1986년 : Illinois Institute of Technology 전산학과 석사
1991년 : Illinois Institute of Technology 전산학과 박사
1992년~1993년 : Governors State University 조교수
19993년~현재 : 경희대학교 전자계산공학과 부교수
<주관심 분야> 실시간 컴퓨팅, 실시간 미들웨어, 멀티미디어 시스템, 시스템 보안

홍 충 선(Choong Seon Hong) 정회원
제 26권 제1호 참조