# 멀티미디어 스트리밍 프레임워크에서 전송 및 세션 관리자의 설계 및 구현

## (Design and Implementation of Transport and Session Managers in a Multimedia Streaming Framework)

임 익 진 † 이 승 룡 †† 정 찬 균 †††
(Eakjin Lim) (Sungyoung Lee) (Changyun Jeong)

**요 약** 본 논문에서는 통합 멀티미디어 스트리밍 프레임워크(Integrated Streaming Service Architecture : ISSA)의 주요 모듈인 전송 및 세션 관리자의 구현에 대한 개발 경험을 소개한다. ISSA는 유니캐스팅/멀티캐스팅 환경의 VOD 시스템과 실시간 방송시스템(라이브캐스팅)과 같은 통합 멀티미디어 스트리밍 서비스 응용을 개발하기 위한 스트리밍 프레임워크이며 RTP/RTCP, RTSP 등의 표준 실시간 전송 프로토콜을 사용함으로써 사용자에게 범용성을 제공한다. ISSA는 다양한 형태의 미디어를 지원하며, 이기종 운영체제와 네트워크에 독립적이며, 실시간 멀티미디어 데이타베이스와 연동하여 사용자에게 데이타베이스 서비스를 제공할 수 있다. 전송 관리자는 ISSA에서 다양한 형식의 멀티미디어 데이타를 알맞게 분해, 조립하는 기능과 패킷화된 미디어 데이타를 실시간으로 전달하는 역할을 수행하며 RTP 프로토콜을 이용하여 구현되었다. 세션 관리자는 미디어 채널의 생성과 제어를 담당하는 기능과 멀티미디어 데이타베이스를 위한 트랜잭션 전송기능을 담당하는 기능을 제공하며 각각 RTSP와 RTTP 프로토콜을 이용하여 구현되었다. 전송관리자와 세션 관리자는 네트워크 인터페이스를 통해 송수신 기능을 수행함으로써 다양한 네트워크 프로그래밍 인터페이스를 지원할 수 있는 유연한 구조를 지니고 있다.

***Abstract*** This paper deals with our experience for the design and implementation of Transport and Session managers in the Integrated Streaming Service Architecture (ISSA) that was developed by the authors. The ISSA is a streaming framework that allows to develop integrated multimedia streaming applications such as VOD system in unicast/multicast and real-time broadcast(live-cast). It also facilities standard real-time transport protocols such as RTP(Real-Time Transport Protocol)/RTCP(Real-time Control Protocol) and RTSP(Real-Time Streaming Protocol) that allows to user openness. The ISSA supports diverse media formats and is independent from underlying networks and operating systems, and compatible with the global real-time multimedia database system (BeeHive) so that streaming media are efficiently retrieved, stored, and serviced. The role of the Transport Manager is to do packetization and depacketization for the different types of multimedia data and delivers the packetized media data in real-time. The Transport Manager uses RTP protocol. The role of the Session Manager is to establish and control the media channel by using RTSP protocol and to deliver the database transactions for the multimedia database by using RTTP(Real-Time Transaction Protocol) protocol. Both the Transport and Session Manager are doing their functions through the network interface in the ISSA that allows developers to various network programming interfaces and provides flexibility to the system.

· 본 논문은 정보통신부 지원 국제공동연구지원사업(과제번호: IJRP- 9803-6)의 지원을 받아 작성되었습니다.
† 비 회 원 : 경희대학교 전자계산공학과
limga@oslab.kyunghee.ac.kr
†† 종신회원 : 경희대학교 전자계산공학과 교수
sylee@oslab.kyunghee.ac.kr
††† 비 회 원 : (주)에스큐브 연구원
cgjeong@oslab.kyunghee.ac.kr
논문접수 : 2000년 1월 7일
심사완료 : 2000년 11월 17일

## 1. 서 론

스트리밍 기술은 데이타 전체를 전송한 후 재생하는 것이 아니라 일부 데이타의 전송 후 즉시 재생하는 방식을 사용함으로서 실시간 특성을 가진 멀티미디어 데이타의 전송에 적합하다. 그러나 지금까지 대부분 스트리밍 시스템에 관한 연구는 다른 시스템과의 연동을 고려하지 않고 독자적인 환경에서 동작하도록 설계되어 이식성과 유연성이 부족하며 추가적인 확장이 쉽지 않다. 그리고 오디오/비디오 코덱 등과 같은 다양한 미디어 처리 기능이 미약하며, 이기종 운영체제 그리고 네트워크 환경을 지원할 수 있는 투명성이 부족하다.

이 같은 제한점을 개선하기 위해서 다른 스트리밍 시스템과의 연동과 확장이 용이할 뿐만 아니라, 다양한 오디오/비디오 미디어 형식을 지원하고 이기종 운영체제와 네트워크 환경에서 동작할 수 있는 적응력을 지닌 통합 멀티미디어 스트리밍 구조인 ISSA를 제안하였다[1][2]. ISSA는 RTSP(Real-Time Streaming Protocol)[6], RTP(Real-Time Transport Protocol)[4][5] 등 표준 실시간 전송 프로토콜을 사용함으로써 범용성을 제공하며, 멀티미디어 실시간 데이타베이스인 BeeHive[3]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. ISSA는 VOD 서비스와 방송 서비스의 세션 제어를 위해 RTSP 등을 지원하는 세션 관리자(Session manager), 멀티미디어의 전송을 위해 RTP, TCP(Transmission Control Protocol), UDP(User Datagram Protocol) 등을 지원하는 전송 관리자(Transport manager), 다양한 미디어 관리를 위한 미디어 관리자(Media manager), CORBA A/V 스트리밍 서비스[9][10]와 같은 다른 스트리밍 서비스 플랫폼과의 연동을 위한 게이트웨이(Gateway) 모듈과 데이타베이스 연결과 트랜잭션을 지원하는 데이타베이스 커넥터(Database connector)로 구성되어 있다.

본 논문에서는 ISSA의 주요 모듈인 전송 및 세션 관리자의 설계와 구현에 대한 개발 경험을 소개한다. 전송 관리자는 ISSA에서 다양한 형식의 멀티미디어 데이타를 알맞게 분해(packetization), 조립(depacketization)하는 기능과, 패킷화된 미디어 데이타를 다양한 네트워크 환경에서 실시간으로 전달하는 역할을 수행한다. 전송 관리자는 실제 데이타를 송수신하는 데 있어 최소한의 오버헤드를 가지고 효율적으로 동작하며, RTP/UDP 와 TCP 프로토콜을 지원함으로써 응용서비스에 매우 유연한 프로토콜 선택이 가능하다. 그리고, 객체지향적으로 설계되어 다른 미디어 전송 프로토콜을 첨가할 수 있어 확장이 용이하며, 컴포넌트 방식으로 다른 응용 프로그램에서 쉽게 사용할 수 있는 특징을 가지고 있다.

세션 관리자는 스트리밍 응용 프로그램 사이의 세션을 새로 생성 또는 소멸하거나 재생 그리고 멈춤 등과 같은 기능을 제공함으로써 스트리밍되는 미디어를 제어할 수 있으며 RTSP를 이용하여 구현된다. 그리고, BeeHive의 트랜잭션을 원격 호스트에서 처리하기 위한 RTTP(Real-Time Transaction Protocol)를 설계하고 구현하였으며, 세션 정보와 컨텐츠 정보의 공유와 분배[7]를 담당하는 SCP(Service Control Protocol) 프로토콜도 제공한다.

제안된 전송 관리자와 세션 관리자는 다양한 시스템에서의 호환성과 확장성을 보장받기 위해 자체적으로 제작된 네트워크 인터페이스라는 네트워크 Wrapper 상에서 구현되어 ISSA 프레임워크를 사용하는 어떠한 네트워크 응용도 간단한 컴파일에 의해 이식을 보장받을 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 전송 및 세션 관리자의 관련연구를 소개하고, 3장에서 통합 분산 스트리밍 프레임워크인 ISSA의 구조와 각 모듈의 기능에 대해서 간단히 살펴본다. 4장에서는 전송관리자, 5장에서는 세션관리자의 설계 및 구현 방법에 대하여 기술하고, 6장에서 구현사례를 소개한뒤, 7장에서 결론을 맺는다.

## 2. 관련 연구

본 장에서는 기존의 스트리밍 시스템에 대한 관련 연구와 전송 및 세션 관리자에 대한 기존의 연구에 대해 살펴보도록 하겠다.

KISS(Communication Infrastructure for Streaming Services)는 독일의 GMD에서 개발한 스트리밍 서비스를 위한 통신 인프라구조로서 네트워크 서비스 응용의 투명한 통합과 실시간 컨텐츠 스트림의 이기종 네트워크 적응을 지원한다[11]. KISS는 데이타속도와 컨텐츠 적응을 위한 SA(Service Applications)와 단말 시스템 사이의 중도적 역할을 수행하는 NAP(Network Access Points)를 사용하여 네트워크 환경의 투명성을 제공한다. 그러나 KISS는 아직 PCM과 MPEG 오디오 형식의 미디어만을 지원하도록 구현되었으며, NAP를 통한 성능 오버헤드를 줄이는 것이 관건이다.

Darwin 스트리밍 서버[13]는 애플사의 오픈 소스 프

로젝트의 하나이며 기본적인 코드는 동사의 QuickTime 스트리밍 서버와 같다. 서버의 동작 환경은 Mac 운영체제의 엑스 서버와 FreeBSD, Linux, Solaris에서 작동하며, QuickTime 파일 형식을 위주로 RTP/RTSP 위에서 서비스를 제공한다. 오픈 소스로서 공개되었지만 동작환경이 범용적으로 사용하는 윈도우를 지원하지 않으며 Quick Time Player에서 지원하는 것과 같은 QuickTime 파일처럼 미디어 서비스의 호환성에 약점을 가지고 있다.

Entera의 TeraCAST서버[14]는 상용 서버로서 1999년에 개발되었다. RTP/RTSP 상에서 대용량의 미디어 시장을 목표로 개발되었으며 서버는 유닉스 계열과 윈도우를 지원하며 Mac 용은 현재 개발 중이다. 클라이언트는 JMF 2.0과 애플의 Quick Time Player를 지원하며 미디어 수용성에 아직 한계를 가지고 있는 것이 단점으로 지적되고 있다.

오픈 소스 Foundation에서 오픈 소스 프로젝트의 일환으로 제작중인 Free Expression 프로젝트[15]는 다양한 전송 프로토콜 위에 RTSP 프로토콜을 사용할 목적으로 개발이 진행중인 스트리밍 서비스 프로젝트이다. 1999년 초에 프로젝트를 시작하였지만 아직 개발 성과가 나오지 않은 상황으로 여러 스트리밍 서버의 역공학(Reverse-Engineering) 작업이 진행중이다. 서버와 클라이언트 사이의 채널에는 RTSP 프로토콜을 사용하였고, 미디어 전송을 담당하는 전송 프로토콜로는 다양한 프로토콜을 서비스하는 계층인 SRM(Scalable Reliable Multicast)에서 담당한다. SRM은 Berkeley의 MASH [16] 프로젝트에서 개발한 전송 계층이다.

## 3. 스트리밍 프레임워크의 기능과 구조

ISSA 프레임워크의 모델은 [그림 1]과 같이 크게 상위 계층의 스트리밍 응용, ISSA와 스트리밍 응용 사이의 인터페이스인 MOA(Media Object Architecture)[12], 데이타베이스 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 분산 되어 있는 컨텐츠의 정보나 세션의 정보를 통합, 분배하는데 있다. 스트리밍 서버의 역할은 모듈화 되어 있는 RTTP 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 연동시에 사용되며, WinAmp의 플러그인은 MP3를 서비스 할 때 사용된다. 또한, 데이타베이스 커넥터는 실시간 멀티미디어 데이타베이스인 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, 데이타베이스의 트랜잭션 요청을 위해 RTTP, 컨텐츠 정보의 분배와 공유를 담당하는 SCP를 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. 자원 관리자는 스트리밍 시스템에서 서비스 품질(Quality of Service: QoS)의 명세화, 매핑, 모니터링, 제어 기능을 제공하며, 메모리 버퍼 관리, 스레드 스케줄링 등의 기능을 수행한다[13] [14].

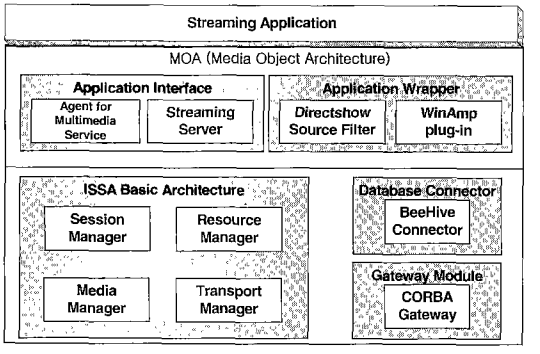


그림 1 통합 스트리밍 서비스 프레임워크의 구성

## 4. 전송관리자 (Transport Manager)

### 4.1 네트워크 인터페이스

ISSA 프레임워크는 다양한 네트워크 환경의 투명성을 제공하기 위하여 네트워크 인터페이스를 통해 네트워크 송수신 기능을 수행한다 [그림 2].

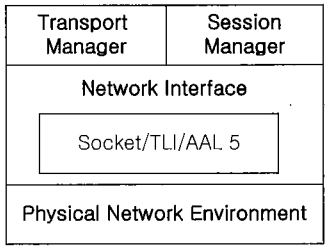


그림 2 네트워크 투명성 제공을 위한 네트워크 인터페이스

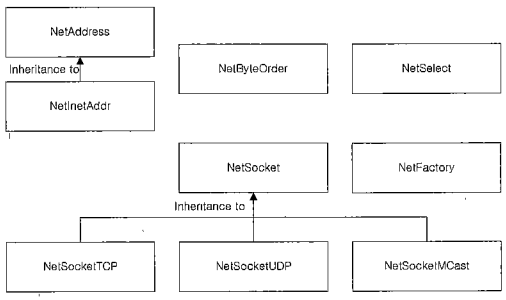


그림 3 네트워크 인터페이스의 UML 클래스 다이어그램

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 소스 코드 레벨에서 이기종간 플랫폼의 호환성도 보장해 준다.

네트워크 인터페이스는 [그림 3]과 같이 구현되었는데, 네트워크 주소를 나타내기 위한 NetAddress 객체와 인터넷 주소를 처리하기 위한 NetInetAddr 객체, 네트워크 바이트의 Byte Alignment를 위한 NetByteOrder 객체, 윈도우와 유닉스를 동일한 이벤트 방식으로 처리하기 위한 NetSelect 객체가 있다. 그리고, NetSocket 객체는 WinSock, 버클리 소켓에서 일관되지 않은 인터페이스를 통합하기 위해 구현되었으며, 여기서 파생된 NetSocketTCP 객체와 NetSocketUDP 객체는 TCP 및 UDP 연결을 NetSocket 객체 함수를 이용하여 구현되었고, NetSocketMCast는 멀티캐스트를 용이하게 처리하기 위해 Time-To-Live 및 구룹 관리 기능이 추가되었다. 그리고, NetFactory 클래스는 네트워크 인터페이스의 소켓 클래스를 생성할 때 사용된다.

### 4.2 전송관리자의 설계

전송관리자는 전술한바와 같이 ISSA에서 다양한 형식의 멀티미디어 데이타를 알맞게 분해, 조립하는 기능과, 패킷화된 미디어 데이타를 다양한 네트워크 환경에서 실시간 전달하는 역할을 수행한다. 즉 전송관리자는 다양한 네트워크 환경에서 여러 종류의 멀티미디어 데이타를 스트리밍 하기 위한 유연하고 확장성 있는 전송 환경을 제공한다. [그림 4]는 전송관리자의 전송 프로토콜 스택으로서 전송관리자에서는 RTP, TCP기반의 다양한 미디어 전송 프로토콜과 IP-유니캐스트 및 IP-멀티캐스트 동작환경을 모두 지원하며, ISSA의 세션관리자에서 제공하는 세션 제어 프로토콜에 관계없이 유연하게 사용할 수 있다.

RTP는 멀티미디어 데이타의 실시간 전송을 지원하는 프로토콜로서, 비디오와 같이 실시간 특성을 가진 멀티미디어 데이타의 종단간 실시간 전달 서비스를 제공한다. 이러한 실시간 전달 서비스는 TCP가 가지고 있는 오버헤드를 효율적으로 극복할 수 있으며, RTP 프로토콜은 패이로드 타입, 소스식별자, 순서 번호, 타임 스탬프, 그리고 제어정보 모니터링을 위한 RTCP등을 이용해 실시간 전달 기능을 지원한다. 일반적으로 RTP는 UDP위에서 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

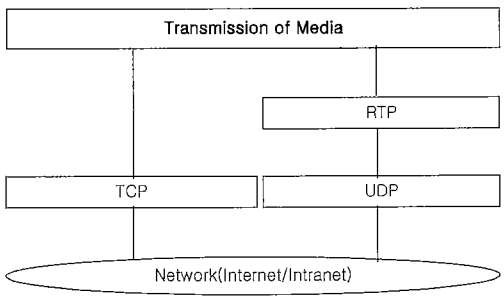


그림 4 전송관리자의 전송프로토콜 스택

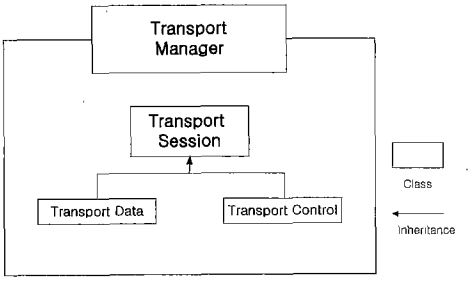


그림 5 전송관리자의 구성

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 세션에서 RTP와 같이 사용되며, QoS 정보를 모니터링하고, 진행중인 세션에 참가한 사용자들의 정보를 전달하는 것을 목적으로 한다. 즉 RTP는 전송기능만을 가지며 RTCP가 세션에 대한 제어 기능을 수행한다. 하지만 RTCP는 최소한의 세션 제어 기능만을 가지고 있다.

전송관리자는 [그림 5]와 같이 논리적으로 크게 전송 세션, 전송 데이타 그리고 전송 제어의 세 부분으로 구성된다. 전송 세션 부분은 전송 세션의 상태, 송수신 통계 등의 전송 세션에 관한 모든 정보를 저장하고, 전송 데이타는 데이타 패킷의 송수신 기능을 수행하며, 마지막으로 전송 제어는 제어 패킷의 송수신 기능을 수행한다.

전송관리자는 위의 세 가지 구성요소를 RTP, TCP 각 전송 프로토콜마다 가지고 있고, 이에 대한 UML 클래스 다이어그램은 [그림 6]과 같으며, 여기에서 전송관리자의 핵심 클래스들은 크게 tmFactory, tmSession, tmDataPacket, tmCtrlPacket 으로 구성되어 있다. tmFactory 클래스는 적절한 미디어 전송 프로토콜(RTP 혹은 TCP)에 해당하는 세션 객체인 tmSession을 생성하는 CreateSession() 오퍼레이션과 데이타 패킷 전송을 위한 tmDataPacket 객체를 생성하는 CreateDataPacket() 오퍼레이션, 그리고 제어 패킷 전송을 위한 tmCtrlPacket 객체를 생성하는 CreateCtrlPacket() 오퍼레이션을 제공한다. tmSession 객체는 현재 생성된 전송 세션에 대한 상태 혹은 통계 정보 등과 같은 모든 정보를 저장하며, 데이타 패킷과 제어 패킷을 보내기 위한 객체인 m_pDataPacket 객체와 m_pCtrlPacket 객체를 저장하고 있다. 이 때 tmSession, tmDataPacket, tmCtrlPacket 클래스는 추상 클래스로서 실제 기능은 RTP 프로토콜의 경우 자식 클래스인 tmRtpSession, tmRtpPacket, tmRtcpPacket에서 수행하며, TCP 프로토콜의 경우 tmTcpSession,

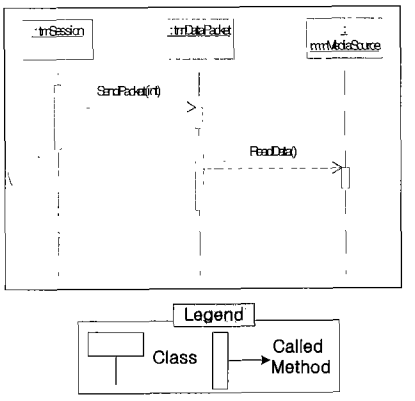


그림 7 데이타 패킷 송신시의 UML 시퀀스 다이어그램

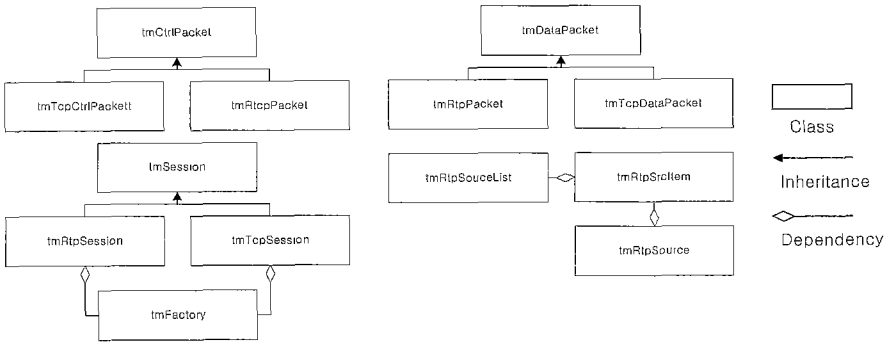


그림 6 전송관리자의 UML 클래스 다이어그램

tmTcpDataPacket, tmTcpCtrlPacket 에서 수행한다. 따라서 새로운 미디어의 전송 프로토콜의 추가시에는 tmSession, tmDataPacket, tmCtrlPacket을 상속하는 새로운 자식 클래스들을 구현하여 주면 쉽게 새로운 전송 구현할 수 있다.

[그림 7]은 데이타 패킷을 송신할 때의 UML 시퀀스 다이어그램으로서, tmSession 객체에서 tmData Packet 객체의 SendPacket() 오퍼레이션을 호출하면 tmDataPacket 객체는 mmMediaSource의 ReadData() 오퍼레이션을 통해 실제 전송할 미디어 스트림 데이타를 얻어와 이를 패킷에 실어 전송한다.

[그림 8]은 데이타 패킷 수신시의 UML 시퀀스 다이어그램으로서, 데이타 패킷이 네트워크를 통해 전송되면 tmSessioin 객체에서 tmDataPacket 객체의 Recv Packet() 오퍼레이션을 호출하고, tmDataPacket 객체는 패킷을 읽어 디코딩 한 후 미디어 스트림 데이타를 mmMediaSink 객체의 WriteData() 오퍼레이션을 통해 mmMediaSink 객체로 넘겨준다.

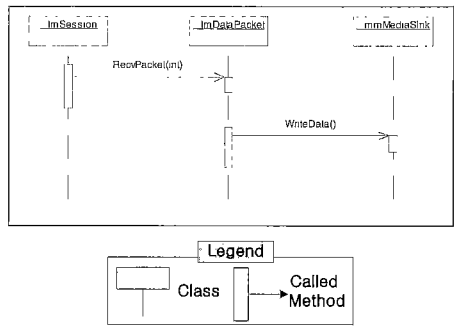


그림 8 데이타 패킷 수신시의 UML 시퀀스 다이어그램

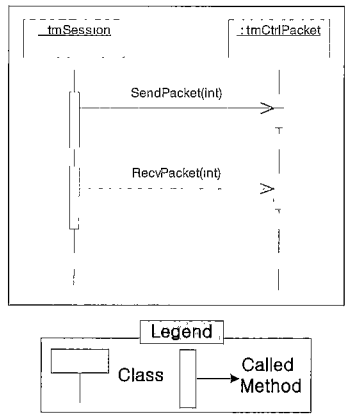


그림 9 제어 패킷 송수신시의 UML 시퀀스 다이어그램

[그림 9]는 제어 패킷의 송수신에 관한 UML 시퀀스 다이어그램으로서, tmSession 객체에서는 제어 패킷 송신 때는 tmCtrlPacket 객체의 SendPacket() 오퍼레이션을 호출하며 수신 때는 RecvPacket() 오퍼레이션을 호출한다.

한편 자체적인 TCP를 이용한 전송프로토콜의 경우 RTP와 같이 미디어 데이타를 효율적으로 전달하기 위해 [그림 10]과 같은 패킷 구조를 가지도록 설계하였다. [그림 10]에서 4 바이트 크기의 media time은 RTP 패킷 헤더의 timestamp와 같이 현재 전달되는 데이타의 타임값을 표현하는 역할을 하며 그 다음 4 바이트의 media data length는 패킷에 실려있는 실제 미디어 데이타의 크기를 나타낸다.

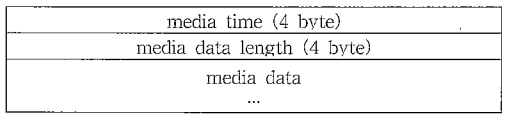


그림 10 TCP 데이타 패킷의 형식

### 4.3 전송관리자의 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 버클리 소켓을 지원하여 하나의 일관된 인터페이스로 네트워크에 대한 접근할 수 있도록 해준다. 또한 타이머의 경우 패킷을 주기적으로 전송하기 위해 필요한 기법으로서, 미디어 데이타 패킷이나 혹은 제어 패킷이 서버 측으로부터 클라이언트 측으로 Push 방식을 이용하여 전송될 때 사용된다. [그림 11]은 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, 데이타를 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 데이타를 얻어올 미디어 소스, 클라이언트는 전달된 데이

타를 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.

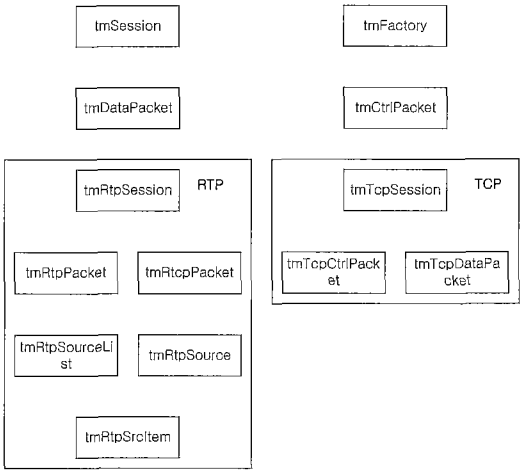


그림 11 전송관리자의 전체 모듈 구성도

## 5. 세션관리자 (Session Manager)

### 5.1 세션관리자의 설계

ISSA의 세션관리자는 스트리밍 응용 프로그램 사이의 세션을 생성, 소멸하거나 재생, 멈춤 등의 기능을 제공함으로써 스트리밍 되는 미디어를 제어 할 목적으로 주로 RTSP를 고려하여 설계되었다. 그러나, 실시간 데이타베이스인 BeeHive와의 연동 과정에서 데이타베이스 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 데이타베이스와 연동되어진 스트리밍 서버의 요청과 그에 대한 처리를 실시간으로 수행하기 위한 프로토콜이기 때문에 실시간 특성을 지닌 데이타 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 SDP(Session Description Protocol)를 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. 다음절에서는 세션 관리자에서 사용된 이러한 프로토콜의 자세한 내용에 대해서 설명하고자 한다.

### 5.2 세션관리자의 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. 세션관리자에서 사용된 표준 프로토콜은 대표적으로 RTSP, SDP등이 있으며, RTTP와 같이 미국 버지니아 대학과 공동으로 제작한 프로토콜과 자체 제작한 SCP도 사용한다.

- **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, 연결지향

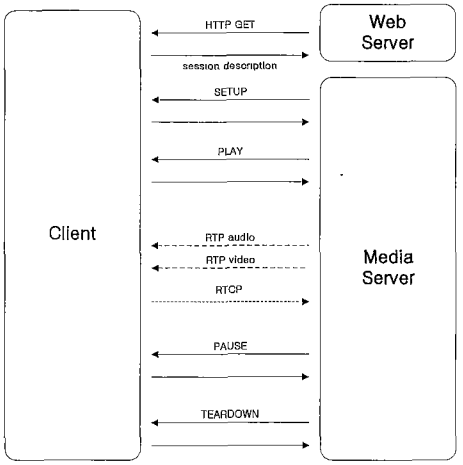


그림 12 RTSP를 사용한 스트리밍의 과정

형(Connection-Oriented) 서비스나 비연결형(Connectionless) 서비스에 상관없이 동작한다. 그리고, 실제적인 미디어의 전송을 담당하는 전송 계층과는 독립적으로 작동한다.

[그림 12]에서는 RTSP를 이용한 스트리밍 과정을 보여준다. RTSP는 세션기술(Session Description)이라는 현재의 세션 정보를 먼저 얻어 오며, 이때는 HTTP(HyperText Transfer Protocol) [8]나 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 문서에의해 표준화 내용이 정의되어 있다. 이는 현재 스트리밍 세션 기술 프로토콜로 사용되고 있으며, 텍스트 기반의 프로토콜이라는 점과 스트리밍 전송 프로토콜과는 독립적이라는 점은 RTSP와 성격면에서 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.

- **RTTP**

RTTP는 ISSA와 BeeHive와의 연동을 위해서 제안된 프로토콜이다. 일반적인 스트리밍 프레임워크에서는 스트리밍 데이타와 메타 데이타를 분리하고, 스트리밍 데이타 부분은 파일 시스템을 사용하여 저장하며, 메타 데이타 부분은 관계형 데이타베이스에 저장된다. 그리고, 파일 시스템과 관계형 데이타베이스를 동시에 사용하는 스트리밍 프레임워크 환경에서는 메타 데이타의 전송에 HTTP와 같은 프로토콜을 사용한다. 하지만, 멀티미디어 데이타베이스를 이용할 경우 스트리밍 데이타(대용량의 멀티미디어 자료)와 메타 데이타(소용량의 관련 정보 자료)는 같은 데이타베이스 내에 존재하며, 이러한 결과 메타 데이타는 스트리밍 데이타와 마찬가지로 실시간 전송을 보장할 수 있는 방법을 가지게 된다. HTTP를 사용하는 메타 데이타 전송 방법은 요청되거나 응답되는 패킷에 타임 스탬프 기능이 없어 멀티미디어 데이타로부터 얻은 실시간 속성을 보장해 주기 어려운 문제가 있다. 따라서, ISSA와 같은 멀티미디어 데이타베이스를 지원해야하는 환경에서는 실시간적인 속성을 갖는 메타 데이타 전송 기능과 스트리밍 데이타의 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 데이타베이스에 요청 시 이루어지게 된다.

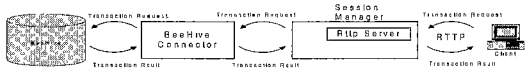


그림 13 트랜잭션 흐름 내에서 실시간 트랜잭션 프로토콜

[그림 13]은 BeeHive와의 연동 시 트랜잭션의 흐름을 보여준다. 클라이언트는 RTTP를 통해 세션 관리자에 요청하는 트랜잭션을 보내게 되면 세션 관리자는 이를 해석하여 BeeHive Connector에 통보하게 된다. 이렇게 전달되어진 데이타베이스의 결과는 다시 BeeHive Connector에서 세션 관리자로 넘어오게 되며, 다시 세션 관리자는 RTTP를 통해 클라이언트에게 결과를 보내게 된다.

[그림 14]는 RTTP의 메시지 형식을 보여주고 있는데 이는 RTSP나 HTTP등과 같은 텍스트 기반의 프로토콜과 유사하게 설계되었음을 알 수 있다. 또한, 프로토콜 자체의 가독성 증대와 파싱에 드는 시간 및 자원의 절약과 개발 기간 단축이라는 장점을 갖는다.

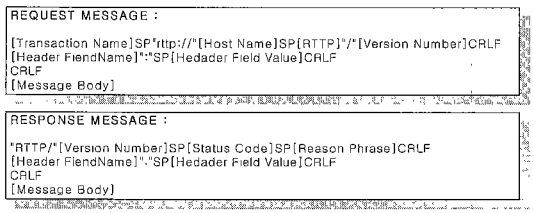


그림 14 RTTP의 메시지 구조

그리고, RTSP에서 사용되었던 여러 헤더 필드 중 패킷의 타임 스탬프등을 그대로 사용할 수 있어 실시간 스케줄링 시 마감시간을 제어할 수 있는 요소로서 사용할 수 있다. 이것은 RTTP의 헤더 필드가 세션에 관련된 사항을 제외하고는 RTSP의 그것을 그대로 사용하였기 때문에 타임 스탬프 이외의 여러 실시간적인 특성을 구현할 수 있는 장점을 그대로 계승하였다고 볼 수 있다. 세션에 관련된 헤더 필드를 제외한 이유는 RTSP는 하나의 세션 관리에 초점을 맞추어 설계된 프로토콜이라는 점과 RTTP는 데이타베이스의 트랜잭션에 실시

간적인 속성을 부여하는데 중점을 둔 프로토콜이라는 차이점 때문이다.

RTTP에 의해서 지원되는 트랜잭션으로는 객체 식별자로 속성값을 얻어올 수 있는 읽기 트랜잭션, 지정하는 객체 식별자로 원하는 속성 값을 저장할 수 있는 쓰기 트랜잭션과 원하는 조건에 맞는 객체들의 정보들을 얻어올 수 있는 찾기 트랜잭션을 지원하며, 마지막으로 데이타베이스에서 미디어 스트리밍을 요청하는 재생 트랜잭션이 있다 [표 1]. 이들은 현재 BeeHive가 지원하는 모든 트랜잭션들이며, BeeHive 프로젝트가 진행됨에 따라 여러 트랜잭션이 추가 될 것으로 예상된다.

표 1 RTTP의 메소드 기능

| 메소드 이름 | 메소드 기능 |
|---|---|
| Read | 객체 식별자, 속성 이름으로 해당 미디어의 정보들을 읽어온다 |
| Write | 객체 식별자와 속성 이름에 지정된 미디어의 정보란에 해당 값을 입력한다. |
| Find | 속성 이름과 Match Expression으로 찾고자 하는 미디어의 객체 식별자를 찾는다. |
| Play | 지정하는 객체 식별자를 갖는 미디어를 데이타베이스에서 스트리밍 하게 한다. |

- **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.

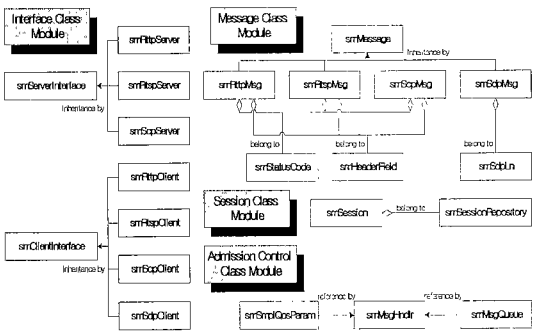


그림 15 세션 관리자의 클래스 다이어그램

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. SCP 메시지의 메소드 기능은 [표2]와 같다.

표 2 SCP의 메소드 기능

| 메소드 이름 | 메소드 기능 |
|---|---|
| GetContentsParam | 서버의 컨텐츠 정보를 받아올 때 사용한다. |
| SetContentsParam | 서버의 컨텐츠 정보를 설정할 때 사용한다. |
| GetChannelsParam | 현재 서비스되는 채널(세션)의 정보를 받아올 때 사용한다. |
| SetChannelsParam | 채널의 생성이나 존재하는 채널의 상태를 바꿀 때 사용한다. |

### 5.3 세션관리자의 구현

세션 관리자는 크게 인터페이스, 메시지, 세션 정보, 승인 제어 클래스 모듈로 나누어진다 [그림 15]. 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 서비스 품질 보장을 위한 승인 제어를 담당하고 있다. 각 클래스 집합에 대한 자세한 설명은 다음절에서 설명한다.

- **인터페이스 클래스 모듈의 구현**

인터페이스 클래스는 [그림 16]과 같이 세션 관리자가 지원하는 프로토콜로 구성되며, 서버 인터페이스 클래스와 클라이언트 인터페이스 클래스로 나누어진다. 서버 인터페이스의 경우 런(Run) 메소드를 통해 초기화 과정이 수행되고, 간단한 송신 및 요청 메소드의 호출로 메시지를 송수신 할 수 있다. 클라이언트 인터페이스의 경우 NewChannel(), FreeChannel()을 통하여 자동적으로 서버와 연결을 이루며 UDP 연결도 통일된 인터페이스를 위해 TCP 연결과 같은 방법으로 동작한다.

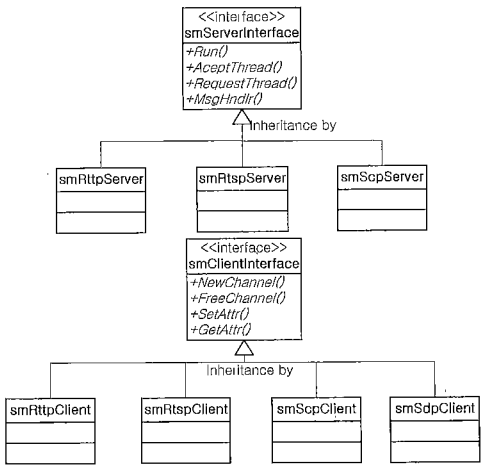


그림 16 인터페이스 클래스 모듈의 클래스 다이어그램

이러한 네트워크를 사용하는 세션관리자의 모듈은 ISSA 프레임워크의 네트워크 인터페이스를 사용함으로써 이기종간의 호환성과 프로토콜의 독립성을 보장받을 수 있다.

[그림 17]은 서버가 가동되었을 때 세션 관리자가 메시지를 받아들이기 위한 준비 과정을 보여준다. 네트워크 인터페이스를 통해 서버와 클라이언트는 네트워크 객체를 생성한 후 서로간의 네트워크 객체를 연결한다.

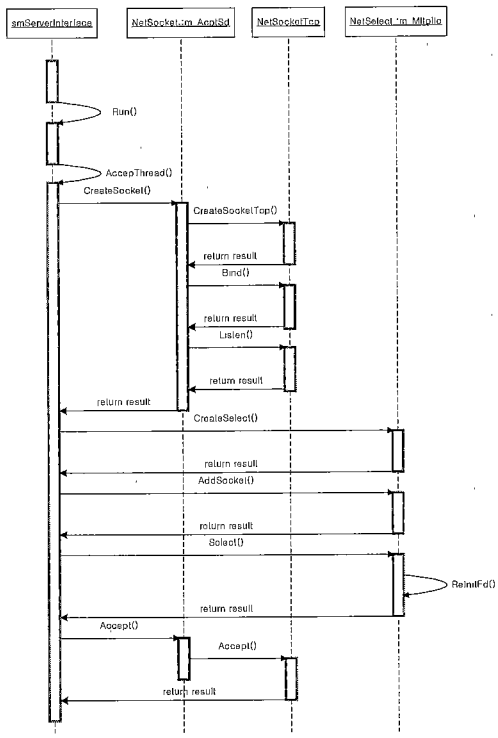


그림 17 서버 인터페이스에서 사용자 요구의 응답 처리 준비를 위한 시퀀스 다이어그램

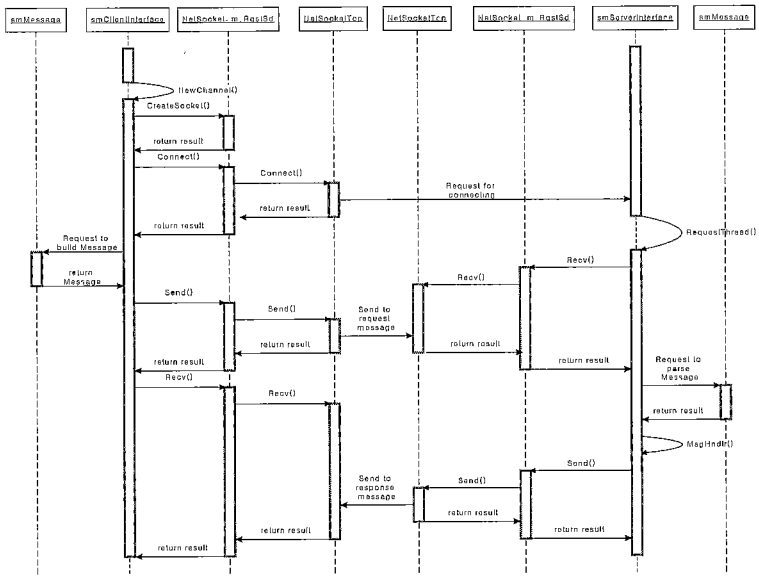


그림 18 클라이언트와 서버 사이의 메시지 전송 및 처리 시퀀스 다이어그램

또한, [그림 18]에서 보듯이 서버 인터페이스와 클라이언트 인터페이스의 요청, 응답에 관한 처리 과정은 서버 인터페이스의 Run()함수에 의해 클라이언트의 접속 요청을 받는 쓰레드가 동작하면서 시작된다. 클라이언트 인터페이스는 NewChannel() 함수를 통해 서버에서 요청을 받기 위해 동작중인 쓰레드에게 내용을 전달한다. AcceptThread의 역할은 클라이언트의 요청을 받기 위한 것이며, 그 후 받아들인 요청의 처리를 위해 작업 쓰레드를 생성한다. 작업 쓰레드인 RequestThread는 클라이언트와 서버사이의 실제 메시지를 주고 받는다. 위와 같은 작업이 끝났을 때 서버 인터페이스는 Logging() 함수를 호출하여 AcceptThread에게 RequestThread의 작업이 끝났음을 통지하고, 클라이언트 인터페이스의 경우 FreeChannel() 함수를 통해 연결을 종료하는 작업을 하게 된다. 한편, 서버 인터페이스 클래스를 상속받은 RTTP, RTSP, SCP 클래스들은 해당 프로토콜의 메쏘드 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.

- **메시지 클래스 모듈의 구현**

메시지 클래스 모듈은 크게 RTSP, RTTP, SCP, SDP 메시지로 나누어진다 [그림 19]. 각 메시지 클래스 모듈은 파서 메소드와 빌드(Build) 메소드를 가지고 있으며, 파서 메소드는 네트워크 단에서 보내진 패킷을 해석하여 클래스안에 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 이름이 따로 들어가며, 응답 클래스의 경우 상태 코드와 Reason Phrase 때문에 세션 상태 클래스를 사용하고 있다. smMessage 클래스는 메시지 클래스의 최상위 클래스로서 추상 클래스로 작성되어 있다.

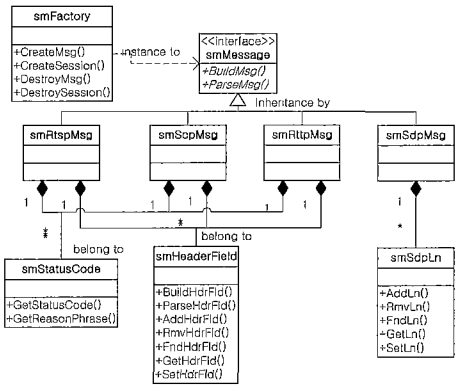


그림 19 메시지 클래스 모듈의 클래스 다이어그램

BuildMsg(), ParseMsg()라는 순가상 함수를 하위 클래스에서 구현하도록 권장하고 있다. smRtspMsg, smScpMsg, smRttpMsg, smSdpMsg는 프로토콜의 메시지를 나타내며, 상태 코드를 smStatusCode, 헤더 필드들을 smHeaderField로 정의하여 사용하고 있다. 그리고, smFactory로부터 메시지의 인스턴스를 받아 사용한다. RTTP는 세션에 관한 정보를 나타내는 헤더 필드를 사용하지 않는다. 이는 RTTP가 세션과는 상관없는 데이타베이스의 트랜잭션을 위해 설계되었기 때문이다.

[그림 20]은 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 필드를 생성하게 된다. 메시지 바디가 있을 경우 BuildMsgBody() 함수를 통해 메시지 바디를 생성한다.

[그림 21]은 서버 인터페이스와 클라이언트 인터페이스에서 메시지를 해석할 때의 시퀀스 다이어그램이다. 메시지를 생성하는 과정과 비슷하지만, 메시지를 해석한

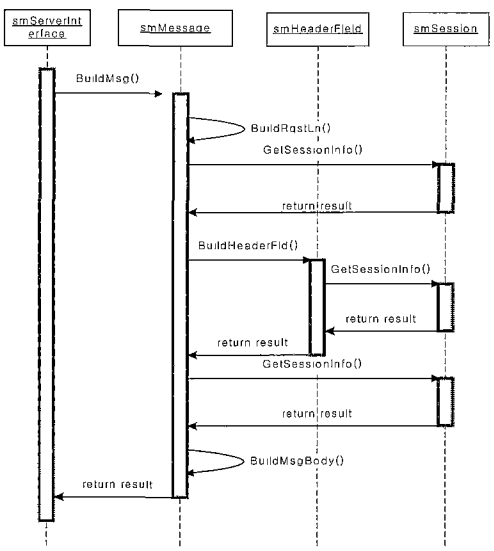


그림 20 메시지 생성 시퀀스 다이어그램

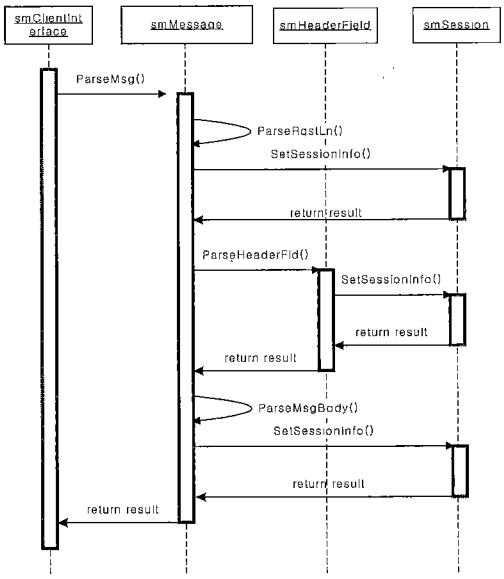


그림 21 메시지 해석 시퀀스 다이어그램

결과를 SetSessionInfo() 함수를 통해 저장하는 것과 메시지 바디의 처리 부분이 메시지 생성의 경우와 다르다.

- **세션 클래스 모듈의 구현**

세션 클래스 모듈은 하나의 세션 정보를 저장하고 있으며, smMessage에서 생성되거나 해석된 내용이 실제적으로 반영되는 클래스로 유니캐스트와 멀티캐스트를 모두 고려하여 설계되었다. 각 smSession의 인스턴스들은 smSessionRepository에 저장되어 관리된다. smSessionRepository는 해쉬 테이블을 사용하여 각 클래스를 저장, 관리하고 있다. 그리고, 내부적으로는 쓰레드 동기화에 의한 데이타의 무결성 보호에 중점을 두어 개발하였다 [그림 22].

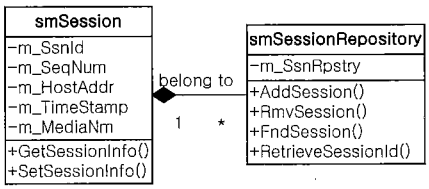


그림 22 세션 클래스 모듈의 클래스 다이어그램

- **승인 제어 클래스 모듈의 구현**

승인 제어 클래스 모듈의 역할은 승인 제어를 통한 서비스 품질의 제공이 목적이며, smMsgHndlr 클래스를 통해 구현된다 [그림 23]. smMsgHndlr 클래스는 자원 관리자에서 현재 시스템이 사용하는 대역폭의 정보를 받아와 smSmplQosParams 클래스에 저장하며, smMsgQueue를 통해 입력받은 처리 요청의 메시지를 smSmplQosParams 클래스의 정보를 판단의 기준으로 삼아 승인 여부를 가리게 된다. 현재의 구현 방법은 해당 미디어의 정적인 요구 대역폭을 통해 승인 제어를 하는 방법과 메시지의 처리 시간을 FIFO를 통해 처리하는 방법으로 구현되었다.

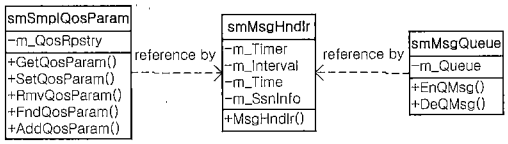


그림 23 승인 제어 클래스 모듈의 클래스 다이어그램

## 6. 구현 사례

본 논문에서 제안한 세션 관리자와 전송 관리자의 구현 사례를 보여 주기 위해서 [그림 24]와 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 24]는 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 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, 음영이 아닌 네모는 스트리밍 플레이어를 통한 세션의 제어를 나타낸다. 스트리밍 흐름은 굵은 선으로 표시되어 있는데, 데이타베이스나 파일 시스템에 존재하는 컨텐츠가 스트리밍 서버를 통하여 클라이언트측에 스트리밍되는 경로를 보여주고 있다.

정보 흐름은 스트리밍 서버가 가동되었을 때 시작되며, 이때 스트리밍 서버 세션 관리자는 스트리밍 서버

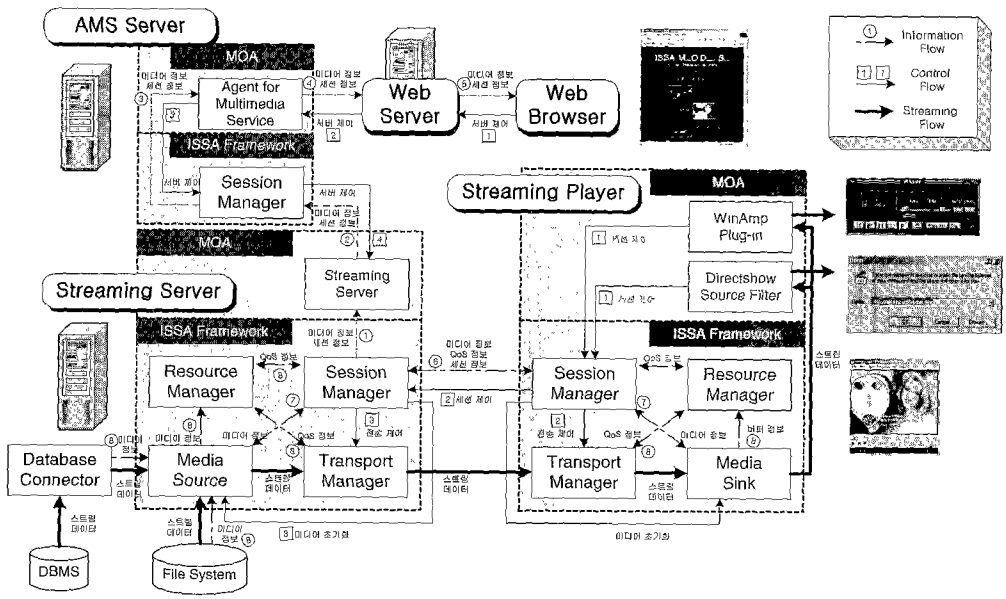


그림 24 스트리밍 서비스에서의 세션 관리자와 전송 관리자의 관계

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 음영이 아닌 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 데이타베이스나 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 투명화 시키고 있음을 뜻한다. 그리고, 이러한 서비스는 멀티캐스트 서비스나 유니캐스트 서비스에 상관없는 유연한 서비스 구조이다.

## 7. 결 론

본 논문에서는 ISSA의 주요 모듈인 전송 및 세션 관리자의 구현에 대한 개발 경험을 소개하였다. 멀티미디어 데이타를 실시간으로 스트리밍하기 위해서는 RTP 프로토콜과 같이 패킷에 미디어의 표현시간, 미디어의 형식, 순서번호 등과 같은 기능을 제공하는 전송 프로토콜이 요구되며, 이같은 요구사항을 반영한 전송관리자는 실제 데이타를 송수신하는 데 있어 최소한의 오버헤드를 가지고 효율적으로 동작하였다. 그리고, RTP/UDP와 TCP 프로토콜을 지원함으로써 응용에게 매우 유연한 프로토콜 선택을 가능케 하였다. 또한, 객체 지향적으로 설계되어 다른 미디어 전송 프로토콜을 첨가하는 등의 확장이 용이하며, 컴포넌트 방식으로 다른 응용 프로그램에서 쉽게 사용할 수 있다.

세션관리자는 세션의 성립, 제어의 기능을 위해 업계에서 표준으로 사용되는 RTSP을 구현하였으며, 스트리밍 서버에서 실시간 데이타베이스의 트랜잭션 처리를 위한 RTTP의 지원과 SCP를 지원하는 명령 서버, 웹 인터페이스의 제공 등 응용의 하위단에서 여러 가지 경로를 통해 스트리밍 서비스를 지원하였다. 이러한 전송관리자와 세션 관리자는 네트워크 인터페이스를 통해 송수신 기능을 수행함으로써 다양한 네트워크 프로그래밍 인터페이스를 지원할 수 있는 유연한 구조를 제공하였다.

향후 연구로는 RTP/UDP의 경우 MPEG과 같이 고대역폭을 요구하는 데이타를 전송할 경우 패킷 손실이

자주 일어날 수 있으므로, 이런 문제를 해결하기 위한 손실된 패킷의 재전송 기법이 요구되며, 패킷 전송 레벨에서의 QoS 지원을 위한 패킷 스케줄러의 필요성이 제기된다. 그리고 세션관리자의 경우, 개발된 여러 기능들을 외부 모듈로 분리해 경량화 해야 할 것이다. 왜냐하면 작은 용량의 세션관리자는 스트리밍 서버보다는 스트리밍 클라이언트에 탑재 될 시에 보다 유용하기 때문이다.

## 참 고 문 헌


[1] 정찬균, 이승룡, "통합 스트리밍 프레임워크의 설계", 제 11회 한국 정보처리학회 춘계 학술발표 논문집, pp. 319-322, 1999년 4월.

[2] Chan-Gyun Jeong, Hyung-Ill Kim, Young-Rae Hong, Eak-Jin Lim, Sungyoung Lee, Jongwon Lee, Byeong-Soo Jeong, Doug-Young Suh, Kyoung-Don 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.

[3] 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.

[4] H. Schulzrinne, "RTP: A Transport Protocol for Real-Time Applications," IETF RFC 1889, Jan. 1996

[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 1890, Apr. 1998

[7] 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

[8] R. Field, UC. Irvine, "HyperText Transfer Protocol - HTTP 1.1," IETF RFC 2068, Jan. 1997

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

[10] 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, January, 1999.

[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.

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

[13] http://www.publicsource.apple.com/projects/streaming/ : Apple Darwin Streaming Server Homepage

[14] http://streaming.entera.com/ : Entera Enterprise Hompage

[15] http://www.free-expression.org/Free-Expression Homepage

[16] http://www-mash.cs.berkeley.edu/mash/Mash Project Homepage



임 익 진

1999년 한신대학교 정보통신학과 졸업. 2001년 경희대학교 전자계산공학과 석사 졸업. 2001년 현재 경희대학교 전자계산공학과 박사과정 재학중. 관심분야는 멀티미디어 스트리밍, 품질 보장, 운영 체제



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



정 찬 균

1998 경희대학교 전자계산공학과 졸업. 2000 경희대학교 전자계산공학과 석사 졸업. 2001년 현재 (주) 에스큐브 연구원 관심분야는 멀티미디어 스트리밍, 보안 네트워크 시스템