# 스트리밍 프레임워크와 멀티미디어 데이타베이스와의 연동기법

# (An Interconnection Method for Streaming Framework and Multimedia Database)

이 재 욱 † 이 승 룡 †† 이 종 원 †††

(Jae-Wook Lee) (Sungyoung Lee) (Jongwon Lee)

**요 약** 본 논문은 실시간 멀티미디어 스트리밍 프레임워크과 멀티미디어 데이타베이스의 연동 모듈인 데이타베이스 커넥터를 소개한다. 스트리밍 시스템과 멀티미디어 데이타베이스를 연동하는 경우 스트리밍 중에도 재생중인 미디어에 관련된 정보들을 데이타베이스로부터 검색 및 재결합이 가능하여 다양한 멀티미디어 데이타베이스 서비스를 제공받을 수 있다. 그러나, 현재 스트리밍 시스템과 데이타베이스와의 연동은 파일 시스템으로 구현되거나, 파일형태의 스트리밍 데이타와 컨텐츠를 다루는 메타 데이타가 분리되어 관리되는 관계형 데이타베이스에 국한되어 있어 다양한 멀티미디어 서비스를 제공하기에 부적합하다. 이런 제약점을 보완하기 위하여 본 논문에서는 스트리밍 프레임워크와 멀티미디어 데이타베이스가 동일한 호스트에 존재한다는 가정 하에 작동되는 IPC 기반의 데이타 베이스 커넥터를 제안한다. 제안된 데이타베이스 커넥터는 데이타베이스 기능을 사용할 수 있도록 읽기, 쓰기, 찾기, 재생 트랜잭션과, 트랜잭션 처리를 위한 인터페이스를 정의하였고, IPC 인터페이스 모듈을 플러그인 형태로 구현하여 본 논문에서 적용한 BeeHive와의 연동뿐 아니라 다른 다양한 멀티미디어 데이타베이스와 연동 시 바로 적용시킬 수 있는 확장성을 가지고 있다. 성능 분석 결과 제안된 IPC 기반 연동기법은 기존의 파일 방식의 연동기법과 비교하여 성능의 저하가 크지 않았다.

**키워드** : 스트리밍시스템, 멀티미디어 데이타베이스, 스트리밍과 멀티미디어 DB 연동, IPC

***Abstract*** This paper describes on our experience of developing the Database Connector as an interconnection method between multimedia database, and the streaming framework. It is possible to support diverse and mature multimedia database services such as retrieval and join operation during the streaming if an interconnection method is provided in between streaming system and multimedia databases. The currently available interconnection schemes, however have mainly used the file systems or the relational databases that are implemented with separated form of meta data, which deals with information of multimedia contents, and streaming data which deals with multimedia data itself. Consequently, existing interconnection mechanisms could not come up with many virtues of multimedia database services during the streaming operation. In order to resolve these drawbacks, we propose a novel scheme for an interconnection between streaming framework and multimedia database, called the Inter-Process Communication (IPC) based Database connector, under the assumption that two systems are located in a same host. We define four transaction primitives; Read, Write, Find, Play, as well as define the interface for transactions that are implemented based on the plug-in, which in consequence can extend to other multimedia databases that will come for some later years. Our simulation study show that performance of the proposed IPC based interconnection scheme is not much far behind compared with that of file systems.

**Key words** : Streaming System, Multimedia Database, Interconnection between Streaming and Multimedia DB, IPC(Interprocess Communication)

† 학생회원 : (주)디지웨이브 기술연구소 연구원
jaeki@oslab.kyunghee.ac.kr
†† 종신회원 : 경희대학교 전자계산공학과 교수
sylee@oslab.kyunghee.ac.kr
††† 비 회 원 : 한국통신데이타(주) 기업부설연구소장
jwlee@ktdata.co.kr
논문접수 : 2001년 9월 10일
심사완료 : 2002년 5월 24일

## 1. 서 론

멀티미디어 운영체제, 통신 및 응용서비스 기술과 이를 지원할 수 있는 고성능 하드웨어의 발전에 힘입어 인터넷 방송, 위성방송, VOD, 무선 이동 환경에서 스트리밍 서비스가 점차 일반화 되어가고 있다. 스트리밍 환경이 성숙되어 감에 따라 스트리밍 시 다양하고 풍부한 멀티미디어 데이타베이스 서비스를 실시간으로 지원 받으려는 요구가 증대되어 가고 있다.

그러나, 기존의 스트리밍 시스템은 주로 스트리밍 서버의 하드디스크에 존재하는 파일을 스트리밍 해주는 구조를 가지고 있거나, 관계형 데이타베이스와 연동하여 멀티미디어 데이타의 정보들은 데이타베이스에 저장하여 사용자들이 멀티미디어 데이타의 정보를 검색할 수 있도록 하고 있다. 관계형 데이타베이스와 연동하는 스트리밍 시스템은 스트리밍 시 서버의 파일 시스템에 존재하는 멀티미디어 데이타를 전송한다. 다시 말하면 멀티미디어 데이타의 정보와 멀티미디어 데이타가 따로 분리되어 관리되어진다. 그리고 스트리밍 시 파일 시스템에 저장되어 있는 멀티미디어 데이타를 스트리밍 하기 때문에 파일 시스템이 가지고 있는 서비스 제약점을 그대로 가지고 있어 검색 및 재결합을 이용한 멀티미디어 서비스를 지원할 수 없다.

이러한 제약점들을 보완하기 위해서는 본 논문에서는 저자가 개발한 실시간 멀티미디어 스트리밍 프레임워크인 ISSA(Integrated Streaming Service Architecture)[1][2][3]와 멀티미디어 데이타베이스와의 연동을 위한 기법으로서 데이타베이스 커넥터를 제시한다. 현재 멀티미디어 데이타베이스 시스템에 대한 연구는 많이 진행중이지만 상용 스트리밍 시스템과 연동을 할 수 있을 정도의 상용 멀티미디어 데이타베이스는 없다. 따라서, 본 논문에서는 학술적으로 연구가 진행되어 소스코드 접근이 가능한 미국 버지니아 대학의 전역 실시간 멀티미디어 데이타베이스인 BeeHive[4][5]를 연동 대상으로 하였다. BeeHive는 실시간, 고장허용, 보안, 멀티미디어 QoS를 크게 강화한 시스템으로 강력한 컨텐츠 관리 기능을 제공한다. 한편, ISSA는 저자가 개발한 스트리밍 시스템 프레임워크로 확장이 용이할 뿐만 아니라, 다양한 오디오/비디오 미디어 형식을 지원하고 이 기종 운영체제와 네트워크 환경에서 동작할 수 있는 적응력을 지니고 있으며 클라이언트 서버 구조를 가진다.

스트리밍 시스템과 멀티미디어 데이타베이스를 연동하면 스트리밍 중에도 재생중인 미디어에 관련된 정보들을 사용자의 요구 사항에 따라 다양한 형태로 검색할 수 있다. 또한, 사용자가 원하는 데이타를 검색, 재결합하여 각 사용자의 요구에 따른 형태로 스트리밍 할 수 있으며, 멀티미디어 데이타의 정보뿐만 아니라 멀티미디어 데이타 또한 데이타베이스의 관리를 받기 때문에 다양한 멀티미디어 서비스를 할 수 있다. 이를 위하여 본 논문에서 다루는 데이타베이스 커넥터에서는 멀티미디어 데이타베이스에서 처리 가능한 트랜잭션을 정의하고, 트랜잭션 처리를 위한 인터페이스를 정의하고 구현하였다. 구현을 위한 응용 분야로 영화 데이타베이스를 선택하였으며, 데이타베이스에서 영화 객체(Movie Object)를 저장하고 다룰 수 있는 읽기(Read), 쓰기(Write), 찾기(Find) 그리고 재생(Play) 트랜잭션을 정의하였다.

ISSA와 BeeHive가 서로 연동하여 작동되는 모델은 트랜잭션 인터페이스와 스트리밍 인터페이스로 나뉜다. 트랜잭션 인터페이스는 BeeHive 트랜잭션을 ISSA 클라이언트에서 요청하고 이를 처리하기 위한 일련의 과정을 다루는 인터페이스이며, 스트리밍 인터페이스는 ISSA 클라이언트와 서버 사이의 미디어 스트림을 처리하기 위한 인터페이스를 의미한다. 그리고 ISSA와 BeeHive간의 멀티미디어 데이타의 정보 및 미디어 스트림, 트랜잭션을 위한 메시지의 교환은 IPC 기반의 공유 메모리와 메시지 큐를 이용한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 소개하고, 3장에서는 ISSA와 BeeHive의 구조와 기능에 대하여 간단히 언급한다. 4장에서는 본 논문에서 제시하는 연동기법으로 IPC 기반의 연동기법에 대해 기술한 뒤, 5장에서는 구현된 IPC 기법과 선행 연구된 연동기법의 성능을 비교하고, 파일 시스템의 경우와 제시한 IPC 기반의 연동기법에 대한 성능을 비교 분석해 본 다음, 마지막으로 6장에서 결론을 맺는다.

## 2. 관련 연구

본 장에서는 스트리밍 시스템과 멀티미디어 데이타베이스의 연동에 관한 관련 연구에 대하여 소개한다.

RealNetwork[6]사의 RealSystem 서버는 RealSystem에서 사용되는 고유의 미디어 포맷을 이용하여 스트리밍 서비스를 한다. 이 미디어 포맷은 네트워크 대역폭을 적게 사용하면서 많은 사용자들에게 스트리밍 서비스를 할 수 있어 RealSystem 사용자의 관리비용을 절감시킬 수 있다. 그리고 RealSystem은 RealSystem SDK라는 개발 툴을 지원하여 RealSystem을 응용한 멀티미디어 시스템의 구축이 가능하며, 전송 방식에서 멀티캐스트 및 유니 캐스트를 모두 지원하고, UDP/IP,

TCP/IP, RTP등의 프로토콜을 지원한다. 그리고 RealSystme은 실시간 세션 프로토콜인 RTSP를 지원하여 각 세션을 관리한다. 이같이, RealSystem은 다양한 네트워크 환경의 지원과 압축률이 높은 멀티미디어 데이타 포맷을 이용한 스트리밍 서비스를 사용하기 때문에 동시에 많은 사용자들에게 양질의 서비스를 제공할 수 있다.

하지만, RealSystem은 멀티미디어 데이타에 관한 정보들을 관계형 데이타베이스에 저장하여 웹을 통해 사용자에게 컨텐츠의 정보를 보여주고 사용자가 선택한 멀티미디어 데이타를 스트리밍 하는 방식을 취한다. 이때 멀티미디어 데이타는 RealSystem 서버의 파일 시스템에 저장되어있다. 또한, 이런 관계형 데이타베이스와의 연동은 멀티미디어 데이타의 정보와 멀티미디어 데이타를 분리하여 관리한다. 이것은 사용자에게 멀티미디어 데이타에 관한 정보를 알려줄 수 있는 장점을 가지고 있으나, 스트리밍 시 결국 파일 시스템에 저장되어 있는 멀티미디어 데이타를 전송하기 때문에 사용자는 두 번의 요청을 하게되므로 그만큼의 오버헤드가 발생한다. 그리고, 파일 시스템에 저장되어 있는 멀티미디어 데이타를 전송하는 것은 파일 시스템이 가지고 있는 제약점을 그대로 가지고 있어 멀티미디어 데이타의 검색 및 재결합을 통한 데이타베이스 서비스를 충분히 활용 할 수 없다는 단점을 가지고 있다.

마이크로소프트사의 멀티미디어 스트리밍 시스템인 NetShow[7]는 라이브 방송 및 오디오, 비디오 데이타 등을 높은 성능으로 전송할 수 있는 기술을 지원하며, 마이크로소프트사의 Microsoft Internet Information Server, Microsoft Windows NT Server 네트워크 운영체제, 인터넷 익스플로러 브라우저와 결합되어 있기 때문에 멀티미디어 스트리밍 시스템 구축에 있어 비용을 줄일 수 있다. 그리고 다양한 네트워크 환경을 지원하며, 멀티캐스트 및 유니 캐스트 기능을 지원하여 UDP/IP, TCP/IP, RTP, HTTP등과 같은 다양한 네트워크 전송 프로토콜을 이용하여 데이타의 전송을 할 수 있고, 소프트웨어 개발 툴(Software Development Kit, SDK)를 지원하여 사용자의 취향에 맞는 멀티미디어 시스템을 구축할 수 있다.

하지만 NetShow는 [그림 1]의 구조에서 보듯이 스트리밍을 위한 멀티미디어 데이타가 파일 시스템에 저장하여 스트리밍을 한다. 이것은 본 논문에서 제시하고 있는 멀티미디어 데이타베이스와의 연동과 달리 파일 시스템에만 의존하고 있는 구조로 파일 시스템이 가지는 단점을 그대로 가지고 있어 멀티미디어 데이타의 검색 및 재결합과 같은 데이타베이스 서비스 기능을 제공하지 못한다.

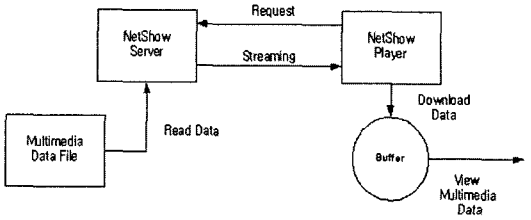


그림 1 NetShow 스트리밍 시스템의 구조

ETRI의 Database Section에서 개발한 BADA-III[8]는 객체지향 자료 모델을 지원하며, 멀티미디어 자료관리, 전문 정보색인 및 검색을 지원하고 데이타베이스 구축 도구의 제공 및 WWW환경과의 통합 지원도구 제공 등의 기능을 가지는 멀티미디어 DBMS로써 멀티미디어 데이타베이스 서비스 및 정보검색 분야로 응용할 수 있다. BADA-III는 2겹 구조의 클라이언트-서버 구조를 가지는 데이타베이스이며, 서버는 자료 저장 시스템과 객체 커널 모듈로 구성되어있으며, 클라이언트는 데이타베이스 응용 프로그램으로 작동하며 클라이언트와 서버 간의 통신은 TCP/IP를 이용한다. 그리고 BADA-III는 객체지향 모델을 지원하는 객체 커널인 DABA-III/OK, BLOB(binary large object), CLOB(character large object) 및 정보 검색 인덱스를 지원하는 자료저장 시스템인 BADA-III/MIDAS, 스키마 관리, 객체관리, 사용자 관리 등을 지원하는 통합된 데이타베이스 관리 도구인 BADA-III/VM, 데이타베이스 응용 프로그램 작성을 위한 C++ 클래스 라이브러리인 BADA-III/C++, WWW와 BADA-III를 연동하기 위한 WWW 통로인 BADA-III/Webgateway로 구성되어있다[그림 2].

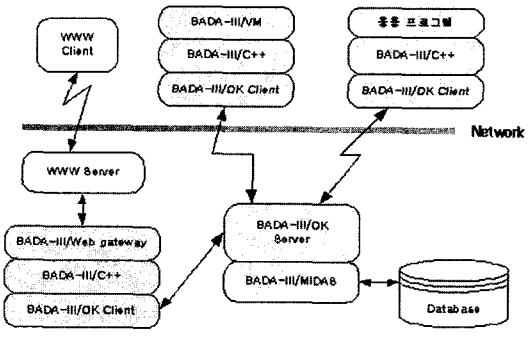


그림 2 BADA-III 구조도

BADA-III는 멀티미디어 데이타를 저장하고 관리하기 용이하도록 설계되었으며, C++클래스 라이브러리를 지원함으로써 다양한 데이타베이스 응용 프로그램을 개발하여 사용할 수 있는 환경을 제공한다. 하지만 이 멀티미디어 데이타베이스는 멀티미디어 데이타를 저장하고 관리하는 기능을 가지고 있지만, 기본적으로 일반적인 자료관리 또는 정보 색인 및 검색을 주된 데이타베이스 서비스로 하고 있어 스트리밍 시스템과의 연동에 그리 적합한 형태를 가지고 있지는 않다. 그러나 두 시스템의 연동을 위한 인터페이스가 존재한다면, 스트리밍 시스템과 연동이 충분히 가능하다고 볼 수 있다.

## 3. ISSA와 BeeHive

이 장에서는 본 논문에서 제시하는 연동 기법을 적용시킨 스트리밍 시스템인 ISSA와 멀티미디어 데이타베이스인 BeeHive 시스템의 기본 구조와 각 시스템의 기능에 관하여 기술한다.

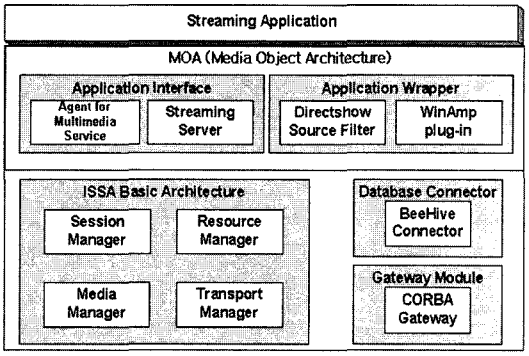


그림 3 통합 스트리밍 서비스 프레임워크의 구성

먼저 ISSA 프레임워크 모델은 [그림 3]과 같이 크게 상위 계층의 스트리밍 응용, ISSA와 스트리밍 응용 사이의 인터페이스인 MOA(Media Object Architecture)[9], 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 분산되어 있는 컨텐츠의 정보나 세션의 정보를 통합, 분배하는데 있다. 스트리밍 서버의 역할은 모듈화되어 있는 RTSP(Real-Time Streaming Protocol)[10] 모듈들의 통합을 통한 미디어의 전송 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 플러그 인은 MP3를 이용한 스트리밍 서비스를 할 때 사용된다. 또한, 데이타베이스 커넥터는 실시간 멀티미디어 데이타베이스인 BeeHive와 연동시켜주는 BeeHive 커넥터, 게이트웨이 모듈은 CORBA[11][12][13] 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 인터페이스, 컨텐츠 정보의 분배와 공유를 담당하는 SCP(Session Control Protocol)를 지원한다. SCP는 데이타베이스 또는 파일 시스템 상에서 작동하는 모든 스트리밍 서버들의 컨텐츠의 정보를 가지고있는 일종의 디렉토리 서버인 AMS와 스트리밍 서버간의 통신을 위하여 정의한 프로토콜로 스트리밍 서버가 새로운 컨텐츠의 추가가 있어서 갱신을 하거나, 서버가 새롭게 작동하여 자신의 컨텐츠 정보를 AMS에게 전송할 때 사용한다. 전송 관리자는 TCP 및 UDP, 그리고 RTP[14]/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 QoS의 명세화, 매핑, 모니터링, 제어 기능을 제공하며, 메모리 버퍼 관리, 쓰레드 스케줄링 등의 기능을 수행한다.

ISSA 프레임워크와 연동하는 BeeHive는 미국 위스콘신 대학에서 개발한 쇼어(SHORE)라는 파일 시스템 위에서 멀티미디어 기능을 지원하는 실시간 멀티미디어 데이타베이스로 실시간 지원, 오디오 및 비디오를 위한 QoS(Quality of Service), 고장허용, 보안 기능을 지원하고, 공개되 시스템이기 때문에 논문에서 제시한 연동 기법을 적용하기에 적당한 시스템이다. BeeHive의 구조는 [그림 4]와 같이 Native BeeHive Sites, BeeHive와 외부시스템과의 연결을 위한 포트, 그리고 외부 시스템들이 BeeHive와 연동을 위한 인터페이스 부분으로 구성되어 있다.

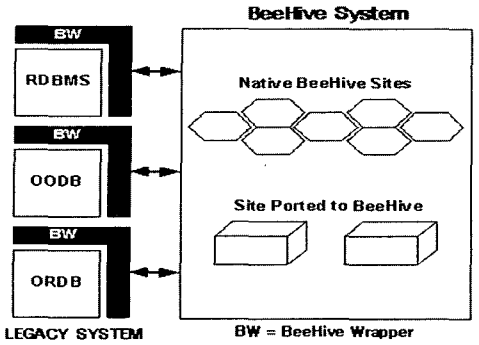


그림 4 BeeHive의 구성

## 4. ISSA와 BeeHive의 연동기법

이 장에서는 본 논문에서 제시하는 연동기법에 관하여 기술한다. 제시하는 연동기법은 모든 멀티미디어 데이타베이스에 적용 가능하도록 인터페이스를 설계하고 구현하였다. 이 인터페이스는 스트리밍 시스템과 멀티미디어 데이타베이스간의 상호 연동 시 필요한 트랜잭션을 처리해주는 트랜잭션 부분과 요청된 멀티미디어 데이타를 전송하는 스트리밍 부분으로 나누어진다[그림 5].

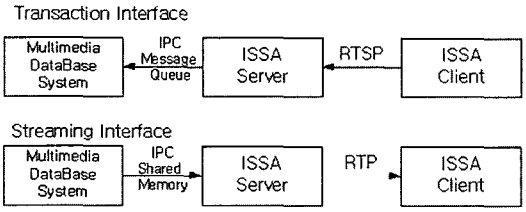


그림 5 스트리밍 시스템과 멀티미디어 데이타베이스 사이의 인터페이스

트랜잭션 인터페이스에서는 BeeHive 내의 트랜잭션을 지원하기 위하여 읽기, 쓰기, 찾기, 재생의 4가지 프리미티브(primitive)를 정의하였으며, 이는 사용자로부터 스트리밍 서버로 요청이 오면 해당 트랜잭션을 멀티미디어 데이타베이스에서 사용되어지는 트랜잭션으로 바꾸어 멀티미디어 데이타베이스에게 데이타를 요청한다. 스트리밍 인터페이스는 요청되어진 트랜잭션에 따라 얻어진 멀티미디어 데이타를 사용자에게 전달해준다. 트랜잭션과 스트리밍 인터페이스를 수행함에 있어 스트리밍 서버와 멀티미디어 데이타베이스를 연동 시켜주는 매개체가 데이타베이스 커넥터이다. 또한, 트랜잭션의 전달을 위해서 IPC 메시지 큐를 이용하며, 메타 데이타 또는 멀티미디어 데이타의 전달을 위해서는 IPC 공유 메모리를 이용한다.

ISSA와 BeeHive 간의 전체 연동 구조도는 [그림 6]과 같다. ISSA 모듈은 BeeHive 커넥터, 미디어 관리자, 전송관리자 모듈로 구성되어 있으며, BeeHive 모듈은 BeeHive 서버와 영화에 관련된 데이타베이스를 관리하는 무비 클라이언트와 BeeHive 서버로 구성되어 있다. 그리고 두 시스템간의 메시지 전달을 위한 IPC 기반의 메시지 큐, 멀티미디어 데이타의 전달을 위한 IPC 공유메모리 블록들로 구성되어 있다. IPC 공유메모리 블록들을 두 개의 시스템이 사용할 때 발생되는 동기화 문제를 해결하기 위하여 메모리 블록의 개수와 동일한 수의 IPC 세마포어를 사용하였다.

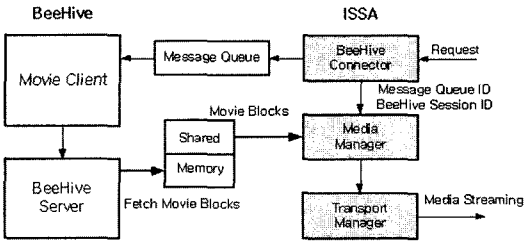


그림 6 ISSA와 BeeHive의 연동 구조도

### 4.1 ISSA 모듈

ISSA와 BeeHive의 연동 시 사용되는 ISSA의 모듈들은 BeeHive와 ISSA 서버간의 세션을 관리해주는 BeeHive 커넥터와 미디어 데이타를 관리해주는 미디어 관리자, 스트리밍 시 미디어 전송을 담당하는 전송관리자로 구성되어 있다.

BeeHive 커넥터는 ISSA와 BeeHive를 연결시켜 주는 매개체 역할을 하는 ISSA 내부의 모듈로, ISSA 클라이언트들이 미디어 스트림을 요구할 때 각 세션의 아이디를 부여하고, BeeHive 무비 클라이언트에게 트랜잭션을 요청하는 메시지 큐의 주소를 얻어오고, 생성된 세션을 삭제하고, 각 세션의 우선순위를 부여하는 역할을 한다. 그리고 MAX_SESSION 이라는 최대 사용자의 수를 설정할 수 있도록 하여 스트리밍 서버의 성능에 따라 최대 사용자 수를 제한할 수 있는 기능을 가지고 있다. 여기서 서버의 성능이라 함은 서버의 CPU 클럭수, 네트워크 대역폭, 메인 메모리 용량 등을 말한다.

[표 1]은 BeeHive 커넥터 모듈의 역할을 수행하는 인터페이스 함수들로, 세션을 생성할 때 메시지 큐를 얻어 오는 *Init()* 함수, 세션을 생성하는 *GetSessionID()*

함수, 세션을 삭제하는 *RemoveSessionID()* 함수, 현재 세션의 우선순위 값을 가져오는 *GetSessionPriority()* 함수 세션의 우선순위를 설정하는 *SetSessionPriority()* 함수로 구성되어 있다.

표 1 BeeHive 커넥터의 함수

```
static int Init( );
static int GetSessionID( int Priority );
static int RemoveSessionID( int ID );
static int GetSessionPriority( int SessionID );
static int SetSessionPriority( int SessionID, int
        Priority );
```

표 2 mmMediaIOBeeHive의 함수

| 함수 원형 | 설명 |
|---|---|
| *int Open(const void *ID, int Flags);* | · 미디어 소스를 오픈한다.<br>· Flag의 상태에 따라 각기 다른 형태로 오픈한다. |
| *int Close();* | · 미디어 소스를 닫는다 |
| *int Read(void *pBuffer, unsigned int Count);* | · 미디어 소스로부터 데이타를 읽는다.<br>· 미디어 소스에서 Count 만큼을 읽어 pBuffer에 저장한다. |
| *long SetPointer (long Offset, int Mode);* | · 오픈된 미디어 소스에서 현재의 포인터 위치를 변화<br>· Offset 만큼 Mode의 위치를 중심으로 이동한다. |
| *int GetTotalLength();* | · 미디어 스트림의 길이를 얻어온다. |
| *void InitAttributes (void);* | · 맴버 변수들을 초기화한다. |
| *int SetMediaInfo (const char *pMediaInfo);* | · BeeHive로부터 미디어 스트림에 대한 정보를 얻어온다. |
| *void ChangeFlag(void);* | · 공유 메모리 사용을 위한 플래그의 상태를 변화시킨다. |

미디어 관리자는 스트리밍 프레임워크 내에서 미디어 스트림이 어떠한 타입의 소스로부터 얻어지며, 그것이 어떠한 종류의 스트림 인가를 판별한다. 그리고, 획득된 미디어의 인코딩 타입을 확인하여 가장 적절하게 처리할 수 있는 방법을 선택하며, 어떠한 미디어 디바이스를 통해 재생되어야 효과적인지를 식별하고 관리하는 모듈이다. BeeHive와의 연동을 위해서 사용되는 미디어 관리자 모듈은 미디어 소스를 선택하고, 읽어오는 역할을 하는 클래스인 mmMediaIO를 상속받은 mmMediaIOBeeHive를 이용한다. 이 클래스는 mmMediaIO의 기본 함수들을 구현하고, BeeHive를 연동하는데 있어 필요한 함수들이 추가되어 있다. mmMediaIOBeeHive의 인터페이스 함수들은 [표 2]와 같다.

전송 관리자는 ISSA에서 네트워크 부분을 담당하는 모듈로써, RTP/UDP 및 TCP 프로토콜을 지원하고 패킷들의 전송 상태를 모니터링 하기 위한 RTCP 프로토콜도 지원한다. BeeHive와의 연동에 있어서 이 모듈의 주된 역할은 미디어 관리자의 mmMediaIOBeeHive로부터 얻어지는 미디어 스트림을 일정한 크기로 패킷 타이징을 하는 역할과 RTP 프로토콜을 이용하여 ISSA 클라이언트에게 전송을 하는 역할을 수행한다.

### 4.2 BeeHive 모듈

BeeHive에서 스트리밍 프레임워트와의 연동에 관련된 모듈은 BeeHive 서버와 BeeHive에서 영화 서비스를 위한 BeeHive 무비 클라이언트로 구성되어있다. BeeHive 서버에는 ISSA 클라이언트에게 전송하기 위한 미디어 스트림이 일정한 크기의 블록으로 나뉘어 저장되어있으며, 미디어 블록의 인덱스 리스트와 미디어 데이타에 관한 정보들이 저장되어있다.

BeeHive 무비 클라이언트는 ISSA와 BeeHive 서버가 연동될 때 메시지들을 주고받을 수 있는 IPC 메시지 큐를 초기화한다. 그리고, BeeHive 서버에게 ISSA로부터 전달되는 메시지들을 BeeHive 서버가 사용하는 트랜잭션으로 변환하여 BeeHive 서버가 공유메모리에 해당 데이타를 넣어주도록 BeeHive에게 요청한다. 다시 말하면 ISSA서버로부터 전달되어진 메시지들을 분석하여, 메시지에 해당하는 트랜잭션으로 변환하고 BeeHive 서버에게 트랜잭션을 요청하는 것이다.

### 4.3 IPC 메시지 큐

스트리밍 시스템과 멀티미디어 데이타베이스와의 연동을 위해서는 서로 통신을 하기 위한 메시지 전달 방법이 필요하며, 이를 수행하기 위한 방법으로 IPC 메시지 큐를 이용하였다. 이 메시지 큐는 IPC 기법을 이용한 것으로 멀티미디어 데이타베이스에서 생성과 초기화를 수행하도록 구현되었다. IPC 메시지 큐의 구조는 [그림 7]과 같다.

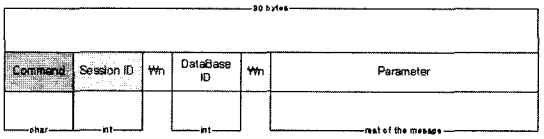


그림 7 IPC 메시지 큐의 구조

*Command* 필드는 메시지의 종류를 나타내는 필드로 각 메시지의 종류는 아래의 [표 3]과 같다. *Session ID* 필드는 클라이언트가 데이타베이스에 저장되어 있는 미

표 3 스트리밍 프레임워크와 멀티미디어 데이타베이스의 연동에서 사용하는 메시지의 종류

| 메시지의 종류 | 설명 |
|---|---|
| *r* | · 요구된 미디어의 정보를 얻어온다. |
| *p* | · 요구된 미디어 블록부터 마지막 미디어 블록까지 데이타베이스로부터 추출하여 전송한다. |
| *q* | · 전송을 중지한다 |
| *l* | · 데이타베이스에 저장된 미디어들의 리스트를 보여준다 |
| *s* | · 데이타베이스에 미디어를 저장한다 |
| *d* | · 데이타베이스에 저장된 미디어를 삭제한다 |

디어 데이타의 요구 시 데이타베이스 커넥터가 부여하는 각 클라이언트의 아이디를 나타내는 부분이다. 이 아이디 값은 공유 메모리와 세마포어를 생성할 때 사용되기도 한다. *DataBase ID* 필드는 미디어 데이타가 데이타베이스에 저장될 때 미디어 데이타에 데이타베이스가 부여하는 고유의 값을 나타내는 부분으로 스트리밍 클라이언트가 선택한 미디어의 아이디 값을 가지는 필드이다. 마지막으로 *Parameter* 필드는 메시지의 종류에 따라 필요한 정보를 넣어주는 필드로써, 예를 들어 재생 트랜잭션일 경우 요구되어진 미디어 스트림의 위치에 따른 미디어 블록의 인덱스 값 또는 다른 스트리밍 시 필요한 정보를 넣은 필드로 메시지의 종류에 따라 다른 용도로 사용된다.

메시지 큐를 이용하여 전달되는 메시지들은 *Command* 필드의 내용에 따라 6가지로 분류된다. 먼저 *r* 메시지는 세션을 생성하고, 세션 아이디를 부여하고, 요청한 미디어의 정보를 얻을 때 사용된다. *r* 메시지로부터 얻어진 세션 아이디는 ISSA의 데이타베이스의 입출력을 담당하는 객체가 생성하는 IPC 공유메모리의 아이디 및 메모리 주소, 그리고 IPC 세마포어의 아이디를 생성할 때 사용된다. *r* 메시지가 데이타베이스에 전달되면 데이타베이스 아이디에 해당하는 멀티미디어 데이타에 관한 정보, 예를 들면 영화의 경우 제목, 감독, 주연 배우 등의 메타 정보들을 얻어 올 수 있다. *p* 메시지는 스트리밍을 위하여 데이타베이스에 저장된 미디어 블록을 요청하는 메시지로, *p* 메시지가 발생하면 요구되어진 데이타베이스 아이디의 미디어 블록을 데이타베이스로부터 추출하여 공유 메모리에 복사하는 작업을 한다. 이 작업은 *q* 메시지가 발생할 때까지 계속한다. *q* 메시지는 멀티미디어 데이타의 전송을 중지하는 메시지로, *q* 메시지가 발생하면 데이타베이스는 미디어 블록을 공유 메모리에 복사하는 작업을 멈추게 된다. *l* 메시지는 현재 데이타베이스에 저장되어 있는 미디어들의 리스트를 보여주는 메시지이다. *s* 메시지는 멀티미디어 데이타를 데이타베이스에 저장하기 위한 메시지로, 이 작업을 통하여 데이타베이스는 저장할 미디어에 고유의 데이타베이스 아이디를 부여한다. *d* 메시지는 데이타베이스에 저장되진 멀티미디어 데이타를 데이타 베이스에서 삭제하는 할 때 사용하는 메시지이다.

[표 4]는 메시지 큐의 인터페이스 함수들과 구조체이다. *QueueMessage*는 메시지의 내용과 메시지의 우선순위로 구성되어 있는 구조체이다. *GetQueue()* 함수는 메시지 큐를 생성할 때 사용하는 함수로, *QueueKey* 파라메타의 값으로 큐의 키 값을 부여하고, *QueueID* 파라메타에 생성된 큐의 아이디를 저장한다. *RemoveQueue()*는 큐의 사용이 모두 끝났을 때 메시지 큐를 삭제하기 위해 사용되는 함수로 파라메타의 종류에 따라 큐의 키 값을 파라메타로 가지는 것과 큐 아이디를 파라메타로 가지는 두 가지의 형태로 분류된다. 그리고 메시지의 전송을 위한 *SendMsg()* 함수와 메시지를 받기 위한 *RecvMsg()* 함수가 있다. *SendMsg()*는 BeeHive와 연동하는 경우 mmMediaIOBeeHive 클래스나 bcBeeHiveConnector 클래스에서 사용하고, *RecvMsg()*는 BeeHive 무비클라이언트에서 메시지 큐에 저장되어있는 메시지들을 꺼내어 갈 때 사용한다.

표 4 IPC 메시지 큐에 관한 인터페이스 함수들과 구조체

```
typedef struct {
  long Priority;
  char Message[MAX_MSG];
} QueueMessage;

int GetQueue( key_t QueueKey, int *QueueID );
int RemoveQueue( key_t QueueKey );
int RemoveQueue( int QueueID );
int SendMsg( int QueueID, QueueMessage *Send );
int RecvMsg( int QueueID, QueueMessage *Recv );
```

### 4.4 IPC 공유 메모리와 IPC 세마포어

스트리밍 시스템과 멀티미디어 데이타베이스와의 연동을 위해 IPC 기법을 이용하는 공유 메모리와 세마포어를 사용하였다. 공유 메모리는 데이타베이스에 존재하는 데이타들을 스트리밍 시스템과 데이타 교환하기 위하여 채택한 방법으로, 데이타베이스에서 꺼내온 데이타를 공유 메모리에 넣으면 공유 메모리에 있는 데이타를 스트리밍 서버가 꺼내어 스트리밍 한다.

일반적인 멀티미디어 데이타의 용량은 작게는 몇 MByte에서 많게는 수십 수백 MByte의 크기를 가지고 있다. 작은 데이타의 경우에는 데이타베이스에 저장할

때 하나의 블록으로 저장하여 한번에 꺼내 줄 수 있겠지만 데이타의 크기가 수십 수백 MByte의 크기라면 한번에 미디어 데이타의 크기만큼 메모리를 할당 할 수는 없다. 이러한 단점을 보완하기 위하여 본 논문에서는 미디어 데이타를 여러 개의 조각으로 나누어 데이타베이스에 저장하고 그 조각들의 인덱스를 저장하는 방식을 사용했다. 또한 멀티미디어 데이타들은 연속적이라는 속성을 가지고 있기 때문에 스트리밍을 하기 위해서는 단일 메모리 블록이 아닌 두 개 이상의 메모리 블록을 필요로 한다. 즉 하나의 메모리 블록은 스트리밍 서버에서 데이타를 읽어 가는 블록이고, 다른 메모리 블록은 스트리밍 서버가 스트리밍 중인 데이타 블록의 다음 데이타 블록을 미리 데이타베이스로부터 추출하여 저장하기 위한 메모리 블록이다. 이런 이유로 본 논문에서는 스트리밍 서버와 멀티미디어 데이타베이스가 데이타를 주고받기 위한 공유 메모리 블록을 두 개로 설정하여 연속적인 속성을 가지는 멀티미디어 데이타를 스트리밍 할 수 있도록 설계하였다.

그리고 두 개의 다른 시스템이 공유 메모리를 사용할 때 동시에 공유 메모리를 엑세스하는 경우와 같은 두 시스템간의 동기화 문제가 발생하는데, 이를 해소하기 위해서 세마포어를 이용하였다. 세마포어는 공유메모리 블록의 개수와 같은 두 개의 세마포어를 생성하고 세마포어의 키 값은 첫 번째 세마포어는 베이스 키 값에 데이타베이스 커넥터가 할당한 세션아이디를 더한 값을 이용하고, 두 번째 세마포어는 베이스 키 값에 세션아이디와 MAX_SESSION값을 더한 값을 이용하였다.

### 4.5 RPC 기반 연동기법

본 논문에서 제안한 IPC기반 연동기법의 선행 연구로 BeeHive 내부에서 사용되고 있는 RPC(Remote Procedure Call) 기법을 이용한 연동기법에 관한 연구를 소개한다. 이 방법은 BeeHive의 내부에서 사용되는 RPC 메소드를 이용하여 ISSA에서 직접 BeeHive 서버를 액세스하는 방법으로 BeeHive와 ISSA간의 모든 통신을 RPC 메소드를 이용하고, ISSA 내부에는 BeeHive

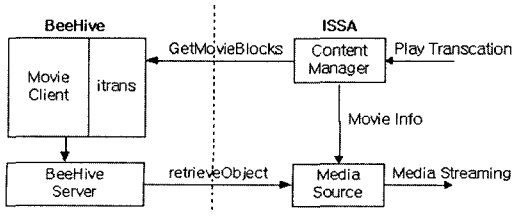


그림 8 RPC기반 ISSA와 BeeHive의 연동 구조

의 RPC 메소드를 랩핑하고 있는 함수들을 선언한 뒤 그 함수들을 사용하여 연동하였다. [그림 8]은 RPC 메소드를 이용하여 ISSA와 BeeHive가 연동되는 구조를 나타낸다.

RPC 기반의 연동기법은 클라이언트로부터 스트리밍 요청이 들어오면 컨텐츠 관리자가 BeeHive 무비 클라이언트에게 미디어의 정보를 요청하여 미디어의 정보를 얻어오고, 미디어 관리자의 미디어 소스 모듈에서 BeeHive 서버에게 RPC 메소드를 이용하여 미디어 블록을 얻어오는 방식으로 본 논문에서 제안한 IPC 방법과 전체적인 구조는 유사하다.

그러나 BeeHive 서버와 ISSA간의 통신 방법과 작동 환경에서 차이가 있다. RPC 기반의 연동에서는 RPC 통신방법을 사용하였고, IPC 기반의 연동기법에서는 IPC 통신방법을 사용한다. 작동 환경에서의 차이점은 RPC는 ISSA와 BeeHive가 동일한 호스트에 존재하지 않아도 작동이 가능하고, IPC는 반듯이 동일한 호스트에 ISSA와 BeeHive가 존재하여야 한다는 것이다. 그리고 RPC 연동기법은 ISSA 내부에서 BeeHive의 표준 기본 메소드를 사용하여 ISSA와 BeeHive가 독립적인 시스템으로써 작동하지 못하며, ISSA와 멀티미디어 데이타베이스의 연동에 있어 BeeHive만을 고려한 연동기법이라는 한계점을 가지고 있어 다양한 멀티미디어 데이타베이스에 적용 가능한 IPC 기법에 비하여 확장성이 떨어진다. 또한, RPC 기법은 분산 환경에서 두 개의 프로그램간의 통신 시 사용하는 방법으로 요청하는 프로그램이 원격 절차의 처리 결과가 반환될 때까지 일시 정지되어야 하는 동기 운영방식이다. RPC 기법은 각 프로그램이 시행하는 원격 절차로 인하여 시간 지연이 발생하기 때문에 스트리밍 시 클라이언트에서 미디어의 끊김 현상이 자주 발생되는 단점을 가지고 있다. 이에 반해 IPC 기반 연동기법은 ISSA와 BeeHive가 완전히 독립적인 형태를 가지고 있기 때문에 BeeHive 뿐만 아니라 다른 멀티미디어 데이타베이스와의 연동이 가능한 확장성을 가지고 있으며, 스트리밍 시 끊김 현상 또한 적어 재생 품질 또한 RPC 기법보다 향상되었다.

### 4.6 IPC 기반 연동기법

이번 절에서는 앞에서 설명한 ISSA와 BeeHive의 연동에 필요한 IPC 기반 구성 요소들이 어떻게 조합되어 스트리밍 시스템과 멀티미디어 데이타베이스가 연동되는지 기술한다.

4.6.1 멀티미디어 데이타베이스 커넥터 모델

제안된 IPC기반 연동 모델은 [그림 9]와 같다. 데이타베이스 커넥터 모듈에서 클라이언트의 요청을 받게

되면 BeeHive 연동 모듈인 bcBeeHiveConnector가 메시지 큐에 저장된 미디어를 요청하고, BeeHive 무비 클라이언트는 BeeHive 서버에게 메시지 큐의 내용을 보내주면, BeeHive 서버는 공유메모리 접근을 위한 플러그인 모듈인 inDataBaseIPC를 이용하여 저장된 미디어를 추출하고 공유 메모리에 넣어준다. ISSA 부분은 공유메모리에 데이타가 저장되면 미디어 관리자의 BeeHive용 mmMediaIOBeeHive 모듈은 공유메모리의 데이타를 전송 관리자에게 전해주고, 전송관리자는 전송된 미디어 데이타를 클라이언트에게 전송하는 구조로 구성되어 있다.

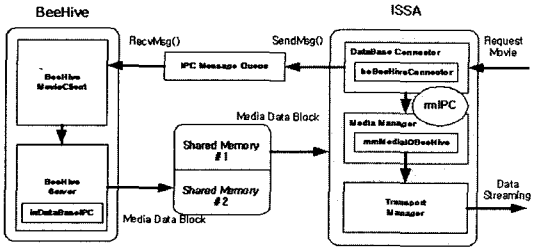


그림 9 ISSA와 BeeHive 연동의 세부 모듈 구조도

[그림 9]와 같은 구조로 두 개의 시스템을 연동하기 위하여 ISSA와 BeeHive 인터페이스에 맞추어 구현된 클래스들은 ISSA 모듈의 bcBeeHiveConnector 클래스와 mmMediaIOBeeHive 클래스, IPC 공유 메모리, 메시지 큐, 세마포어를 사용하기 위한 클래스인 rmIPC 클래스를 구현하였다. 그리고 BeeHive에서 IPC 공유 메모리, 메시지 큐, 세마포어와 데이타베이스에 접근하여 데이타를 얻어오는 inDataBaseIPC 클래스를 구현하였다. 각 클래스의 연관 관계를 나타낸 클래스 다이어그램은 [그림 10]와 [그림 11] 이다.

bcBeeHiveConnector 클래스는 BeeHive와 연동을 위한 데이타베이스 커넥터의 초기화 및 BeeHive와의 세션을 생성 및 소멸하는 역할을 담당한다. mmMediaIOBeeHive 클래스는 ISSA의 미디어 관리자에서 미디어의 입력을 담당하는 추상 클래스인 mmMediaIO 클래스를 상속받아 BeeHive와의 연동 시 BeeHive에 저장되어 있는 미디어를 열고, 읽어오는 역할을 한다. rmIPC 클래스는 데이타베이스와의 연동을 위한 IPC 공유 메모리와 IPC 세마포어를 생성, 관리하고, BeeHive에게 메시지를 보내주는 메소드를 가지고 있는 클래스이다. inDataBaseIPC 클래스는 데이타베이스에서 IPC 공유 메모리를 접근하고, 미디어 블록을 데이타베이스로부터 추출하는 인터페이스를 정의한 클래스로 스트리밍 시스템과 연동 시 모든 데이타베이스의 작업은 이 클래스를 통하여 이루어진다.

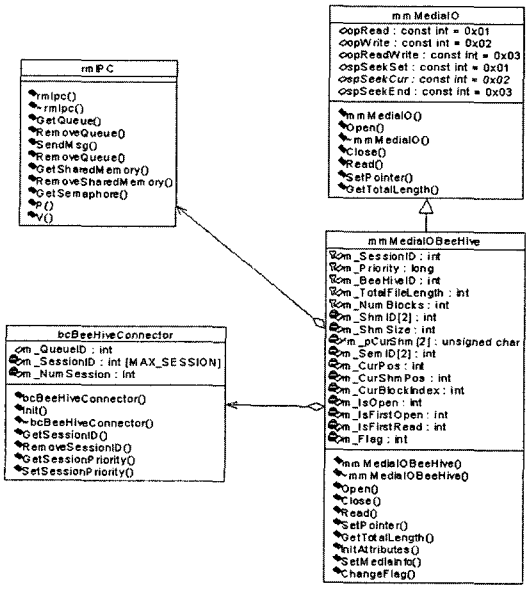


그림 10 BeeHive 연동을 위한 ISSA의 세부모듈 클래스 다이어그램

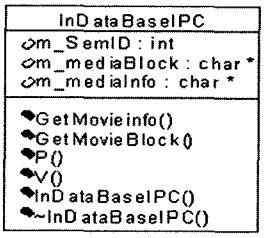


그림 11 데이타베이스 연동을 위한 인터페이스 클래스 다이어그램

4.6.2 데이타베이스 커넥터 작동 알고리즘

IPC 기반 연동모델 작동 알고리즘은 [표 5]에서 보여주는데, 이는 ISSA 클라이언트로부터 BeeHive에 저장되어 있는 멀티미디어 데이타의 스트리밍이 요구될 때 ISSA와 BeeHive의 세부 모듈의 작동을 의사코드로 나타낸 것이다.

ISSA의 클라이언트로부터 BeeHive에 저장되어 있는 멀티미디어 데이타의 스트리밍 요청이 들어오면(*Line 1*), BeeHive 커넥터를 초기화하고 메시지 큐의 주소를 얻어

표 5 ISSA와 BeeHive 간의 연동 의사 코드

```
Line  1: ISSA Server receives the Multimedia
         Streaming Request from the ISSA Clients
Line  2: Initialize the BeeHive Connector and
         Get the Message Queue address from the
         BeeHive MoiveClient
Line  3: Create mmMediaIOBeeHive Object
Line  4: Send r Message to the BeeHive MovieClient
         // r message Creates session and requests
         the Media Information for the multimedia
         streaming //
Line  5: Create Shared Memory and Semaphore
Line  6: Get Shared Memory address and Semaphore
         key Value
Line  7: Send p Message to the BeeHive MoiveClient
         // p message requests the invocation of
         streaming Start //
Line  8: While( Requested Media Block != End of
         Media Block ) {
Line  9: BeeHive Server copies Media Block to
         Shared Memory
Line 10: ISSA Server streams Media Block from
         Shared Memory
         }
Line 11: Send q Message to the BeeHive MovieClient
         // q message requests streaming stop //
```

온다(*Line 2*). ISSA는 스트리밍을 위한 mmMedia IOBeeHive 객체를 생성한다(*Line 3*). mmMediaI OBeeHive 객체는 BeeHive 무비클라이언트에게 *r* 메시지를 보낸다. 이 메시지를 보냄으로써 데이타베이스와의 연동을 위한 세션을 생성하고, 무비 클라이언트에게 미디어의 정보를 요청한다(*Line 4*). IPC 공유 메모리와 세마포어를 생성한다(*Line 5*). mmMediaIOBeeHive 객체는 공유 메모리의 주소와 세마포어의 키 값을 얻어온다(*Line 6*). mmMediaIOBeeHive 객체는 BeeHive 무비 클라이언트에게 *p* 메시지를 보낸다. 이 메시지를 이용하여 스트리밍의 시작을 요청한다(*Line 7*). BeeHive 서버는 inDataBaseIPC 객체를 통하여 미디어 데이타를 추출하고, 추출된 첫 번째 미디어 데이타를 공유 메모리 블록에 복사한 후 공유 메모리 블록의 제어권을 mmMediaI OBeeHive에게 넘겨준다. 제어권을 넘겨받은 mmMediaI OBeeHive는 공유 메모리의 내용을 스트리밍 하기 시작한다. 첫 번째 미디어 블록이 스트리밍 되는 동안 BeeHive 서버는 다른 공유 메모리 블록에 다음에 스트리밍 할 두 번째 미디어 블록을 복사한다. mm MediaIOBeeHIve는 현재 스트리밍 하고있는 공유 메모리블록의 데이타가 떨어지면 다음 공유 메모리 블록의 제어권을 넘겨받아 스트리밍을 계속한다. 이러한 BeeHive 서버와 mmMediaIOBeeHive 객체간의 작업은 스트리밍을 하고 있는 미디어의 마지막 미디어 블록이 BeeHive 서버 추출되어 스트리밍 될 때 가지 계속된다 (*Line 8,9,10*). 선택된 미디어의 모든 미디어 블록이 클라이언트에게 전송을 마치면 mmMediaIOBeeHive 객체는 *q* 메시지를 발생하여 스트리밍의 종료를 요청한다 (*Line 11*).

IPC 기반 연동 알고리즘은 세션의 생성 및 초기화, 스트리밍, 세션 종료의 세 가지 단계로 분류할 수 있다.

• 세션 생성 및 초기화

세션의 생성 및 초기화 단계는 메시지 큐의 주소를 가져오고, 세션의 아이디를 부여하고, 생성된 세션의 IPC 공유 메모리와 세마포어를 할당하는 과정이다. 세션 생성 및 초기화 과정의 시퀀스 다이어그램은 [그림 12]와 같다. ISSA 클라이언트에서 BeeHive 데이타베이스에 저장되어진 미디어 요청하면 bcBeeHiveConnector 클래스의 *Init()* 함수를 호출하여 BeeHive 커넥터를 초기화하고, *GetQueue()* 함수를 호출하여 rmIpc 객체로부터 메시지 큐의 주소를 얻어온다. ISSA의 미디어 관리자는 mmMediaIOBeeHive 객체를 생성하고, *GetSessionID()* 함수를 호출하여 BeeHive 커넥터로부터 세션 아이디를 부여받는다. 그리고 mmMediaIOBeeHive 객체는 *Get SharedMemory()* 함수와 *GetSemaphore()* 함수를 이용하여 공유 메모리와 세마포어를 각 두 개씩 생성하고 공유 메모리와 세마포어의 아이디를 얻어온다. 이것으로 ISSA와 BeeHive를 연동하여 스트리밍을 하기 위한 초기화 과정을 마친다.

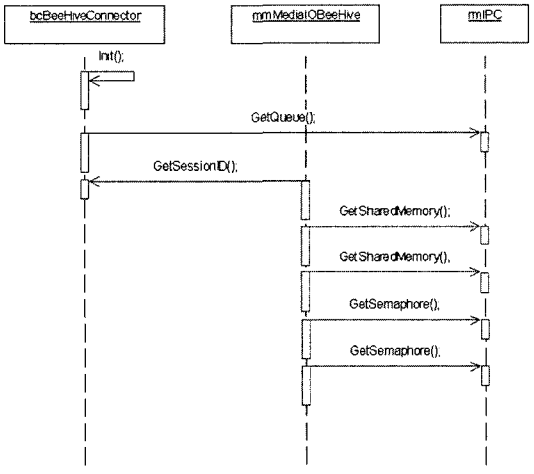


그림 12 세션 생성 및 초기화 시퀀스 다이어그램

• 미디어 데이타 스트리밍 Operation

미디어 데이타 스트리밍 Operation은 미디어 데이타의 오픈과정과 실제적인 스트리밍의 두 가지 단계로 나누어 설명할 수 있다. 먼저 미디어 데이타의 오픈과정은 [그림 13]의 시퀀스 다이어그램에서 보여주는 바와 같이 미디어 관리자의 미디어 소스에서 mmMediaIOBeeHive 객체의 *Open()* 함수를 호출한다. *Open()* 함수가 호출되면 mmMediaIOBeeHive 객체는 IPC 메시지 큐에 *r* 메시지를 *SendMsg()* 함수를 이용하여 전달한다. 메시지가 전달된 BeeHive 무비 클라이언트는 BeeHive 서버에게 *r* 메시지에 해당하는 트랜잭션을 발생한다. 그런 다음 BeeHive 서버는 *GetMovieInfo()* 함수를 호출하여 요청된 미디어의 정보를 추출하고, *P()* 함수를 이용하여 공유 메모리 첫 번째 블록의 제어권을 가져온 후 *memcpy()* 함수를 이용하여 공유 메모리에 미디어 정보를 복사한다. 복사가 끝나면 *V()* 함수를 이용하여 공유 메모리 블록의 제어권을 mmMediaIOBeeHive 객체에게 넘겨주고, mmMediaIOBeeHive 객체는 *P()* 함수를 호출하여 공유 메모리 블록의 제어권을 넘겨받은 후, *memcpy()* 함수를 이용하여 공유 메모리 블록의 내용을 얻어온다. BeeHive 서버로부터 얻어온 미디어 데이타의 정보는 *SetMediaInfo()* 함수를 이용하여 mmMediaIOBeeHive 객체의 멤버 변수에 저장되고 요청된 미디어 데이타의 오픈 과정을 종료된다.

미디어 데이타의 오픈 단계가 종료되면 실질적인 스트리밍 과정이 시작된다. 스트리밍 과정은 미디어 관리자의 미디어 소스가 mmMediaIOBeeHive 객체의 *Read()* 함수를 호출하면서 시작된다. *Read()* 함수가 호출되면 *Read()* 함수는 *SendMsg()* 함수를 이용하여 *p* 메시지를 IPC 메시지 큐에 전달한다. BeeHive는 inDataBaseIPC 객체를 이용하여 rmIPC 객체의 *RecvMsg()* 함수를 호출하고, 메시지 큐의 메시지를 받아온다. 그 다음 요청된 미디어의 미디어 블록을 *GetMovieblock()* 함수를 이용하여 가져온다. 그리고 *P()* 함수를 호출하여 IPC 공유 메모리 블록의 제어권을 가져오고, 공유메모리 블록에 추출된 미디어 블록을 복사한 후 *V()* 함수를 호출하여 공유메모리 블록에 대한 제어권을 포기한다. 그런 다음 계속해서 두 번째 공유 메모리 블록에 다음 미디어 블록을 같은 방식으로 복사한다. mmMediaIOBeeHive 객체는 BeeHive 서버가 제어권을 포기한 공유 메모리 블록의 미디어 데이타를 스트리밍 하기 시작한다. 첫 번째 공유 메모리 블록에 저장된 데이타 블록이 스트리밍 되는 동안 BeeHive 서버는 두 번째 공유 메모리 블록에 다음 데이타 블록을 복사한다. 그리고, 첫 번째 공유 메모리의 데이타 블록이 모두 소진되면 스트리밍 서버는 BeeHive 서버가 가지고 온 두 번째 공유 메모리 블록의 데이타를 스트리밍 한

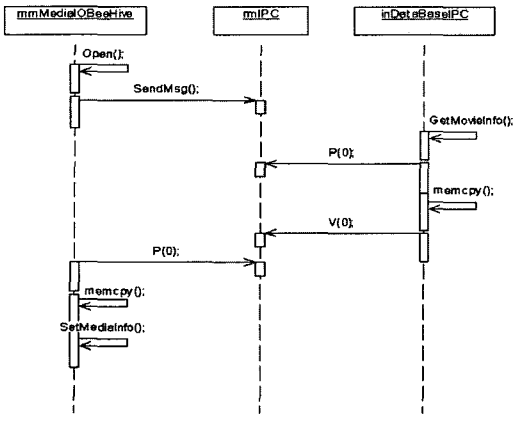


그림 13 미디어 데이타 Open 시퀀스 다이어 그램

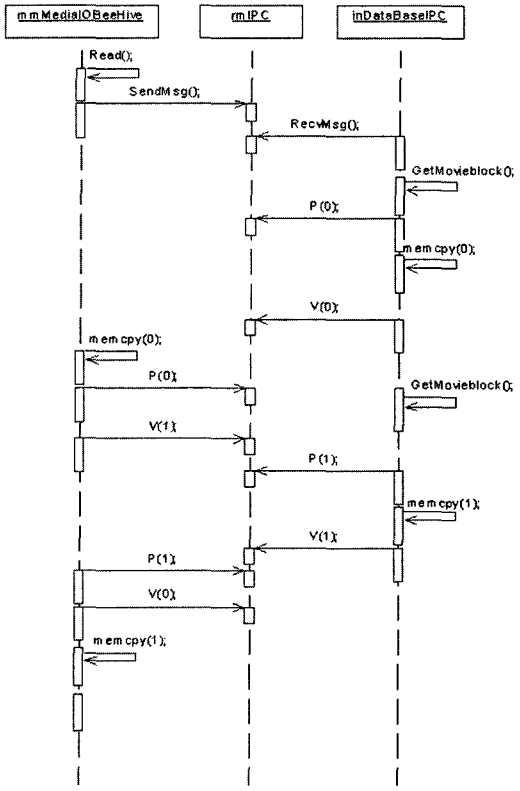


그림 14 미디어 데이타 스트리밍 시퀀스 다이어 그램

다. 이러한 과정은 스트리밍 될 마지막 데이타 블록이 공유 메모리 블록에 복사될 때까지 반복된다. [그림 14]는 이 같은 스트리밍 과정을 보여주는 시퀀스 다이어그램이다.

• 세션의 종료

세션의 종료과정은 세션을 닫고 할당받았던 공유 메모리와 세마포어를 제거하는 과정이다. 마지막 데이타 블록의 스트리밍 끝나면 mmMediaIOBeeHive 객체의 *Close()* 함수가 호출되고, *Close()* 함수에서 rmIPC 객체의 *RemoveSharedMemory()*와 *RemoveSemaphore()* 함수를 호출하여 공유 메모리와 세마포어를 삭제한다. 그리고 미디어 관리자의 미디어 소스는 mmMediaI OBeeHive 객체를 소멸시킴으로써 BeeHive 서버와의 연동 세션을 종료하게 된다.

## 5. 성능 평가

성능 평가 실험 환경은 10Mbps의 대역폭을 가지는 인트라넷 환경 및 인터넷 환경에서 수행되었다. Bee Hive와 ISSA 서버는 SunOS 5.5.1이 탑재된 SUN 워크스테이션에서 작동되었으며, BeeHive에 저장된 미디어 데이타는 두 서버 시스템간의 데이타 교환을 하기 위하여 생성하는 공유 메모리 블록들이 시스템에 주는 부하가 적고, 스트리밍 서비스 품질에 영향이 적은 512KByte 크기로 분할하여 저장하였다. 클라이언트는 ISSA에서 개발된 독자적인 RTP 소스필터를 적용시킨 마이크로소프트사의 Windows 미디어 플레이어 6.4가 사용되었다. 성능 평가는 기존에 스트리밍 시스템이 채택하고 있는 파일 시스템에 저장되어진 미디어 데이타를 스트리밍 했을 때와 멀티미디어 데이타베이스에 저장되어진 미디어 데이타를 IPC 기반 연동기법을 사용하여 스트리밍 하였을 때를 서버에서 클라이언트에게 전송하는 초당 데이타 전송률을 비교함으로써 멀티미디어 데이타베이스에 저장된 미디어 데이타를 추출하여 스트리밍 하는 것이 파일 시스템에 저장되어 있는 미디어 데이타를 스트리밍 하는 것과 어떠한 차이가 있는 지 비교하였다. 본 성능평가에서 제시하는 전송하는 데이타의 양은 OSI 7 레이어의 어플리케이션 레이어에서 사용하는 데이타 크기 말하는 것이며, 전송 패킷의 크기는 미디어가 인코딩 되어있는 인코딩 비트레이트에 따라 그 크기가 정해진다.

본 논문에서는 기존의 스트리밍 시스템들이 채택하고 있는 파일 시스템에 저장된 미디어를 스트리밍 할 때와 멀티미디어 데이타베이스에 저장된 미디어를 스트리밍 할 때 각각의 경우에 따른 데이타 전송률을 비교하여 그 성능 차이를 평가 해보았다. 이 실험은 인트라넷과 인터넷의 두 환경에서 이루어졌으며 다른 스트리밍 시스템들과의 비교를 위하여 일반적인 스트리밍 시스템 환경을 가상으로 구축하여 실험을 진행하였다. 그리고 네트워크 대역폭이 시간에 따라 변하는 것을 고려하여 동일한 시간에 같은 호스트에 데이타베이스용 스트리밍 서버와 파일 시스템용 스트리밍 서버를 작동 시켰다. 각 그래프들의 가로축은 스트리밍 경과 시간(Sec.)을 나타내고, 세로축은 초당 전송 데이타 양(BytePerSecond)을 나타낸다. 그리고 이 실험에서 또한 1392640bits/sec로 인코딩 되어진 MPEG-1 미디어를 사용하였다.

실험 1은 인트라넷환경에서 스트리밍 서버가 클라이언트에서 전송하는 미디어 데이타 양을 측정한 결과이고[그림 16], 실험 2는 인터넷 환경에서 실험한 결과[그림 17]를 나타낸다.

실험 1과 2를 통하여 인트라넷, 인터넷 환경에서 스트리밍 서버가 클라이언트에게 전송하는 데이타 전송량을

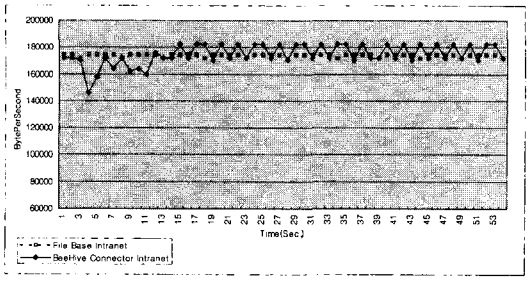


그림 16 실험 1: 인트라넷 환경에서의 멀티미디어 데이타베이스와 파일 시스템 이용 시 스트리밍 서버의 평균 전송률 차이

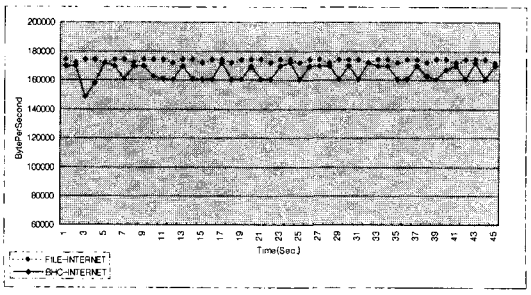


그림 17 실험 2: 인터넷 환경에서의 멀티미디어 데이타베이스와 파일 시스템 이용 시 스트리밍 서버의 평균 전송률 차이

알 수 있었고, 스트리밍 서버는 네트워크 환경에 상관없이 미디어 인코딩 비트레이트 만큼의 데이타를 전송한다는 것을 알 수 있었다. 실험 결과 파일 시스템에 저장되어진 미디어를 스트리밍의 경우는 일정한 양의 데이타를 시간에 관계없이 스트리밍 서버가 전송할 수 있도록 데이타를 공급해주며, IPC기반의 연동기법을 이용하여 BeeHive와 연동한 스트리밍 서버의 경우는 데이타베이스로부터 미디어 블록을 추출하는 데 걸리는 초기 지연으로 인하여 스트리밍 초기에 어느 정도 시간이 지난 후에 안정적인 스트리밍 한다는 것을 알 수 있었다. 그리고 멀티미디어 데이타베이스와 연동하여 스트리밍 하는 경우 파일 스트리밍과 달리 데이타 전송 양의 변화 폭이 자주 생기는 이유는 스트리밍 서버가 두 개의 공유 메모리 블록에서 데이타를 꺼내올 때 메모리 블록을 교체하면서 생기는 지연 시간으로 인하여 발생하는 것이다. 이런 데이타 전송 양의 변화 폭은 스트리밍 시크게 영향을 끼치는 요소는 아니다.

따라서, 제안한 데이타베이스 기반의 스트리밍 방법이 파일 시스템 기반의 데이타를 스트리밍 하는 것에 비하여 큰 성능에 저하 없이 멀티미디어 데이타베이스 서비스를 제공할 수 있는 장점을 보여주었다. 향후, 두 개의 공유 메모리 블록을 사용하는 데에서 기인되는 오버헤드를 줄이기 위하여 캐쉬 기능을 적용한다면 데이타 전송 양의 변화 폭을 줄일 수 있어 성능 향상을 이룰 수 있을 것으로 예측된다.

## 6. 결 론

스트리밍 프레임워크와 멀티미디어 데이타베이스와의 연동은 기존의 스트리밍 시스템이 가지고 있는 한계점을 극복하여 다양한 스트리밍 서비스를 할 수 있다는 장점을 가지고 있다. 기존의 스트리밍 시스템은 서버의 파일 시스템에 저장되어있는 미디어 데이타를 스트리밍 하는 방식을 채택하고 있으며, 데이타베이스와의 연동에 있어서는 자신이 저장하고 있는 미디어 데이타의 정보만을 관계형 데이타베이스에 저장하여 관리하고 있다. 이러한 방법은 파일 시스템에 저장되어있는 데이타를 스트리밍 하기 때문에 미디어 데이타의 검색 및 재결합을 이용한 스트리밍을 할 수 없는 제약점을 가지고 있다. 이와는 달리, 스트리밍 시 미디어 데이타가 멀티미디어 데이타베이스의 관리를 받게 되면, 사용자의 취향에 맞는 멀티미디어 데이타를 검색하고 재결합하여 스트리밍 하는 것이 가능하므로 데이타베이스 서비스를 받을 수 있어 파일 시스템이 제공하는 서비스의 제약점을 극복할 수 있다.

이러한 점에 착안하여 본 논문에서는 스트리밍 시스템인 ISSA와 실시간 멀티미디어 데이타베이스인 BeeHive의 연동 모듈로써 IPC 기반의 데이타 베이스 커넥터를 제안하였다. 이 연동 모듈은 멀티미디어 데이타베이스의 트랜잭션을 처리하기 위한 트랜잭션 인터페이스와 멀티미디어 데이타베이스에 저장된 미디어 데이타를 스트리밍 하기 위한 스트리밍 인터페이스를 정의하였고, 그리고 두 개의 시스템간의 메시지 및 데이타의 전달을 위하여 IPC 기반의 메시지 큐, 공유 메모리, 세마포어를 구현하였다.

성능 평가 결과 본 논문에서 제시한 IPC 기법을 이용하여 데이타베이스에 저장된 미디어 데이타를 스트리밍 하는 것이 기존의 파일 시스템에 저장되어있는 미디어 데이타를 스트리밍 하는 것에 비하여 스트리밍 서비스 품질이 나쁘지 않다는 것을 알 수 있었다.

제안된 연동기법은 IPC 인터페이스 모듈을 플러그인 형태로 구현하였기 때문에 향후 다른 멀티미디어 데이타베이스와 연동 시 적용시킬 수 있는 확장성을 가지고 있으며, 향후 다양한 멀티미디어 서비스를 위해서 필요한 기법이라고 할 수 있다.

향후 연구로는 두 개의 공유 메모리 블록을 사용하면서 발생되는 오버헤드를 줄이기 위해 캐쉬 기능을 공유 메모리에 적용하는 방법에 대한 연구를 수행할 예정이다.

## 참 고 문 헌

[1] 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.

[2] 임익진, 이승룡, 정찬균, 멀티미디어 스트리밍 프레임워크에서 전송 및 세션 관리자의 설계 및 구현, 한국정보과학회 논문지, 제7권 제1호, 2001년 2월, pp.24-37.

[3] 이재욱, 이승룡, 홍인기, 스트리밍 프레임워크에서 미디어 관리자의 설계 및 구현, 정보과학회 논문지, 2001년 6월, pp.273-287

[2] 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.

[3] J. Stankovic, S. Son and J. Liebeherr, "BeeHive: Global Multimedia Database Support for Dependable, Real-Time Applications," In Proc. of Second Workshop on Active Real-Time Databases, Lake Como, Italy, September 1997.
[4] RealNetworks, ReslSystem G2, http://www.realnetworks.com.
[5] Microsoft, Windows Media Technologies, http://www.microsoft.com/netshow.
[6] BADA-III Multimedia DBMS, http://bada.etri.re.kr/team/bada/bada3/bada3.html
[7] 김형일, 이승룡, "멀티미디어 QoS를 위한 미디어 객체 구조의 설계", '98 한국정보과학회 춘계 학술발표 논문집, 1998년 4월, pp.699-701.
[8] H. Schulzrinne, A. Rao, R. Lanphier, Real-Time Streaming Protocol(RTSP), IETF RFC 2326, April 1998.
[9] 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.
[10] Object Management Group, Control and Management of A/V Streams specification, OMG Document telecom/97-05-07 ed., October 1997.
[11] 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.
[12] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, IETF RFC 1889, January 1996.

이 재 욱
2000년 한신대학교 정보통신학과 졸업. 2002년 경희대학교 전자계산공학과 석사. 2002년 현재 (주)디지웨이브 기술 연구소 연구원. 관심분야는 멀티미디어 시스템, 스트리밍 시스템

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 교수. 관심분야는 실시간 컴퓨팅, 실시간 미들웨어, 멀티미디어 시스템

이 종 원
1981년 전북대학교 수학과 졸업. 1986년 Illinois Institute of Technology 전산학 석사. 1991년 ~ 1998년 한국통신 멀티미디어연구소 연구원. 1999년 ~ 현재 한국통신데이타(주) 기업부설연구소장. 관심분야는 실시간 시스템, 데이타베이스 시스템, 지리정보시스템