MediaADE: MHEG 기반 멀티미디어/하이퍼미디어 응용 개발 환경

(MediaADE: A MHEG-based Multimedia/Hypermedia Application Development Environment)

이 세 훈† 왕 창 종††

(Sei-hoon Lee) (Chang-jong Wang)

요 약 : 본 논문에서는 멀티미디어/하이퍼미디어(M/H) 응용 개발 환경인 MediaADE(MHEG-based multimedia/hypermedia Application Development Environment)를 설계 및 구현하였다. MediaADE는 계층적 구조로서 각 계층간에는 인터페이스가 존재하여 계층간 독립성 및 이식성을 극대화할 수 있다. 각 계층은 실행, 추상화, 저장 계층으로 구성되며, 실행 계층과 추상화 계층간에는 프로그래밍 인터페이스가, 추상화 계층과 저장 계층간에는 네트워크/데이터베이스 인터페이스가 존재한다. MediaADE는 MHEG를 사용하는 개방형 M/H 응용 개발 플랫폼이다. 추상화 계층은 MHEG 엔진으로서 M/H 객체 관리, 시공간 동기화 처리, 인코딩/디코딩 및 링크/액션 처리 기능을 포함하며, 프로그래밍 인터페이스는 M/H 객체를 조작할 수 있도록 DLL 형태를 가지며, 네트워크/데이터베이스 인터페이스는 ODBC와 표준 통신 프로토콜을 지원한다. 따라서, 실행 계층은 응용 개발자가 MediaADE에서 제공하는 API를 이용하여 개발한 응용이 되며, 저장 계층은 개발 환경에 따라 ODBC를 지원하는 데이터베이스 관리 시스템이 된다. MediaADE는 초고속 통신망상에서 M/H 응용을 개발하기 위한 개발 환경으로 이용될 수 있을 것이다.

Abstract : In this paper, we design and implement a multimedia/hypermedia application development environment, MediaADE(Mheg-based multimedia/hypermedia Application Development Environment). The MediaADE has layered structure for the open system, and there are interfaces between each layer. The structure is composed of the runtime, abstract, and storage layer. There are two interfaces in the MediaADE structure as follows: API between runtime and abstract layer, and network/database interface between abstract and storage layer. The MediaADE uses the MHEG for representing multimedia/hypermedia information. The abstract layer is a MHEG engine that includes the object handler and synchronization controller in order to interpret the objects efficiently. The API is supported as dynamic linking library, and the network/database interface supports the ODBC(Open DataBase Connectivity) and standard communication protocol. Therefore, the runtime layer is an application that is developed using the API of MediaADE. And the storage layer is a DBMS that supports the ODBC depends on the developing environment. The MediaADE will be used as an environment for M/H applications development on Information Superhighway.

1. 서 론

초고속 통신망의 대두와 컴퓨터의 고성능화, 소프트웨어 기술의 발전에 따라 분산 환경에서의 멀티미디어/하이퍼미디어(M/H : Multimedia/Hypermedia) 응용들이 현실화되었으며, 이는 기존의 응용 시스템 환경에 커다란 영향을 줌과 동시에 더불어 새로운 응용 분야를 만들어 냈다[1].

분산 M/H 응용 분야는 원격 교육 [2, 3, 4], 컴퓨터 지원 공동 작업[5, 6], 멀티미디어 회의 시스템 및 멀티미디어 그룹웨어[7], 멀티미디어 정보 시스템[8, 9], 주문형 비디오[10] 등이 있다. M/H 응용 개발을 위한 기반 기술로 개발 환경에 대한 플랫폼에 대한 연구가 필요하며[11], 응용 프로그래밍 인터페이스(API:Application Programming Interface) 형태의 개발 환경이 제공되어야 할 것이다. 이러한 네트워크 기반의 M/H 응용 개발 환경은 여러 가지 이질적인 환경에 응용할 수 있는 개방형 구조를 가지고 있어야 하며, 이질적인 환경에서 M/H 정보의 상호 교환을 지원할 수 있는 표준 형식과 응용 프로그램 개발을 쉽게 할 수 있는 프로그래밍 인터페이스와 네트워크/데이터베이스 인터페이스를 제공해야 한다. M/H 응용 개발 환경에 대한 연구로는 DAVE[12], MuX[13], MADE[14]등이 있으

나, 분산된 이질적 환경에서 M/H 응용을 쉽게 개발할 수 있는 개발 환경과 응용간에 상호 교환할 수 있도록 표준화된 저장방식을 따르고 있지 않다.

M/H 표준화에 관한 연구는 국제 표준화 기구인 ISO/IEC와 기업에서 수행되고 있는데 표준화 분야로는 멀티미디어의 코딩, 정보 표현, 분산 환경을 위한 표준 등으로 분리되어 진행되고 있다[1, 7]. 멀티미디어 데이터의 압축 · 복원 표준은 JPEG, MPEG, H.261(px64), DVI등이 대표적이며, 하이퍼미디어 문서 구조에 대한 표준은 SGML/HyTime, ODA/HyperODA등이 있다. 프리젠테이션 처리를 위한 표준으로는 PREMO가 있으며, M/H 정보 상호교환 형식 표준은 MHEG, OMFI, SMSL, QMF, AVI등이 있다.

MHEG(Multimedia and Hypermedia information coding Expert Group)[15]은 네트워크 기반의 상호 작용 형태의 응용에 적합한 가장 포괄적인 M/H 정보 표현 및 상호교환 형식 표준으로서, 네트워크를 통해 정보가 교환될 수 있도록 M/H 정보의 표현, 시공간 동기화, 상호 교환, 실시간 전송, 최종 형태 표현 등을 포함한 인코딩/디코딩을 정의하고 있다. MHEG은 객체 지향적인 방법을 도입하여 하이퍼미디어 내의 모든 객체를 하나의 클래스로부터 상속받도록 함으로써 일관적인 구조를 갖는 멀티미디어 객체들의 집합을 정의하고 있다. 또한 여러 가지 미디어 형태들을 교환할 수 있는 기능을 제공하는데, 미디어 데이터들은 JPEG, MPEG등 기타 국제 표준에 따라 부호화된다. MHEG을 이용하기 위해서는 MHEG 엔진이 필요한데, 표준에서는 이러한 MHEG엔진의 기능만을 간단히 기술하고 있을 뿐이며 개발자가 환경에 맞게 개발하여 사용하도록 하고 있다. MHEG 객체로 표현된 M/H 정보는 MHEG 엔진을 통해 인코딩/디코딩 및 프리젠테이션 처리를 위한 객체 해석, 그리고 사용자와의 상호 작용 등을 하며, MHEG 표준에 기반한 어떠한 응용도 엔진을 필요로 한다. 이러한 MHEG 엔진에 관한 연구로는 Umass Lowell[16], Mannheim[17]등이 있으며, MHEG에 기반한 응용으로는 공공 네트워크를 통한 VOD등의 IMS1[10], 정보 검색 시스템인 HIRS[18], Link Server[19]등이 있다.

이러한 MHEG에 기반한 분산 M/H 응용을 개별적으로 개발한다면 많은 시간과 비용이 소요되며 중복 투자되는 부분이 많게 된다. 따라서 응용에서 공통적으로 필요한 부분과 보다 경제적으로 개발할 수 있는 체계적인 개발 환경이 필요하다. 이러한 개발 환경을 이용하는 것은 생산성을 높일 수 있는 방법이 될 수 있을 것이다. 분산 M/H 응용의 요구가 증대될 수록 개발 환경에 대한 연구는 매우 중요하며 파급 효과가 클 것이다.

따라서 본 논문에서는 멀티미디어/하이퍼미디어 응용 개발을 위한 개방형 환경인 MediaADE (MHEG-based multimedia/hypermedia Application Development Environment)를 설계 및 구현한다. MediaADE는 M/H 정보 표현 형식으로서 MHEG을 사용하며, 각 계층간에는 인터페이스가 존재하여 계층간 독립성 및 이식성을 극대화할 수 있는 개방형 구조이며 각 계층은 실행, 추상화, 저장 계층으로 구성되고, 실행 계층과 추상화 계층간에는 프로그래밍 인터페이스가, 추상화 계층과 저장 계층간에는 네트워크/데이터베이스 인터페이스가 존재한다. 저장 계층은 M/H 정보를 저장하는 계층으로서 인터페이스를 통해 요구되는 데이터를 관리한다. 추상화 계층은 M/H 정보를 해석하는 핵심적 계층이다. 추상화 계층은 정보를 해석하기 위한 MHEG 엔진으로서 MHEG 객체를 효율적으로 관리할 수 있는 객체 처리 및 시공간 동기화 처리 구조를 포함하고 있다. 실행 계층은 사용자와 상호 작용하는 M/H 응용으로서 프로그래밍 인터페이스를 통해 추상화 계층의 내부 구조에 대한 지식 없이도 개발될 수 있다. MediaADE의 전체 환경 구성은 클라이언트/서버로서 클라이언트에 추상화 계층과 실행 계층을 두고, 서버에 저장 계층을 둔다. 이러한 환경은 서버의 부담을 최소화하고 클라이언트에서 다양한 응용을 개발할 수 있는 형태이다. 프로그래밍 인터페이스는 MHEG 객체를 조작할 수 있는 DLL(Dynamic Linking Library) 형태의 응용 프로그래밍 인터페이스이며, 네트워크/데이터베이스 인터페이스는 ODBC(Open DataBase Connectivity)와 표준 통신 프로토콜을 지원한다. 따라서, 실행 계층은 응용 개발자가 MediaADE에서 제공하는 API를 이용하여 개발한 응용이 되며, 저장 계층은 개발 환경에 따라 ODBC를 지원하는 데이터베이스 관리 시스템이 된다.

이 논문은 2장에서 기존의 M/H 응용 개발 환경과 MHEG 표준을 분석하고, 3장에서 MediaADE를 설계 및 구현, 4장에서 결론과 향후 연구 방향을

서술한다.

2. 멀티미디어/하이퍼미디어 응용 개발 환경 및 표준

본 장에서는 기존 멀티미디어/하이퍼미디어 응용 개발 환경에 대한 연구와 표준에 대해서 비교 분석한다.

2.1 분산 멀티미디어/하이퍼미디어 응용 개발 환경

초고속 통신망에 대한 관심과 함께 분산 환경에서의 M/H 응용 개발이 활발히 진행되고 있으며, 이러한 M/H 응용을 효율적으로 개발하기 위해 응용 개발 환경에 관한 연구가 시작되고 있다. 본 절에서는 이러한 연구들을 고찰함으로서, 응용 개발 환경이 가져야 할 기능을 분석한다.

DAVE(Distributed Audio Video Environment)[12]는 중앙 집중화된 형태나 분산된 응용 프로그램 개발 지원과 함께 API를 지원한다. 또한 디바이스와 미디어 확장성을 제공하고, 객체 재사용성을 증진시키며, 상호 동작과 네트워크 독립성을 지원한다. 이는 응용 프로그램 개발자가 분산 환경의 응용 프로그램을 쉽게 개발할 수 있도록 해주며, 재사용 가능한 멀티미디어 툴킷 생성을 가능하게 한다. DAVE는 UNIX상에서 IP 프로토콜을 사용하여 개발되었다. 객체지향 분석, 설계 방법을 채택하여 디바이스의 세부적인 내용을 고도로 추상화(abstraction)시킴으로써, 상속과 데이터 독립성을 유지하여 개발자가 새로운 디바이스를 쉽게 추가할 수 있다.

ETRI에서 개발한 MuX(Multimedia cross)[13]는 각종 분산형 멀티미디어 응용 프로그램이 요구하는 멀티미디어 정보의 처리와 입출력을 위한 기본 함수(API)를 제공하고, 이런 함수의 수행을 위한 확장된 운영체제를 지원하여 주는 멀티미디어 전용 서버이다. MuX는 운영체제가 제공하지 못하는 멀티미디어, 네트워크에 관련된 서비스를 API 형태로 사용자에게 제공한다. 응용 프로그래머는 MuX에서 제공되는 객체 지향 특성을 갖는 API를 사용하여 프로그램을 작성하게 된다. 객체 지향적 특성을 지니기 때문에 마이크, 스피커, 카메라, 화면 등의 객체를 생성하고, 이들을 연결하고, 초기화시키고, 플레이시키는 것이 응용 프로그래머가 해야할 일이 된다. 실제로 각 객체를 메모리에 생성하고 처리하는 것은 운영체제의 차원에서 MuX가 제공한다. 화상 전화와 같은 응용 프로그램은 약간의(100 라인 이하) 코딩만으로 작성이 가능하다. MuX는 객체 지향 방법을 도입하였고, 오디오, 비디오 데이터에 대해서는 완벽한 실시간 통신 및 실행을 지원한다.

유럽 공동체의 ESPRIT Ⅲ 프로젝트의 일환으로 개발된 MADE(Multimedia Application Development Environment)[14]는 다양한 종류의 멀티미디어 데이터를 하드웨어에 무관하게 처리할 수 있는 통일된 방법을 제공하며, UNIX, MS-Windows 환경을 지원한다. MADE는 멀티미디어 응용 프로그램 개발의 위한 객체지향 라이브러리를 제공한다. 모든 멀티미디어 객체 편집을 위한 각각의 편집기를 제공하고 있고, 멀티미디어 데이터 처리시 중요한 동기화 처리를 위한 두가지 방법을 제공하고 있다. 그리고, 현재까지는 MADE만의 고유한 데이터 표현 포맷(MIFF)을 사용하고 있지만, Hytime이나 MHEG과 같은 데이터 표현 표준에 대해 쉽게 이식할 수 있도록 고려하고 있다.

그러나, 이들 대부분은 멀티미디어 데이터 표현에 어떤 표준도 따르고 있지 않으며, 개방형 구조를 취하고 있지 않다. 따라서, 표준 멀티미디어 데이터 표현 방법에 기반을 두고, 응용 프로그래머가 분산 M/H 응용을 쉽게 개발하도록 하는 API와 이러한 API를 수행해 주는 플랫폼에 대한 연구가 요구된다.

M/H 응용 개발 환경을 위한 요구 사항들로는 다음과 같은 것들이 있다.

첫째, 데이터의 재사용성을 높이기 위하여 이질적인 시스템간의 M/H 데이터 상호 교환을 위한 표준 표현 방식에 기반을 두어야 한다.

둘째, 멀티미디어 데이터간의 시공간 동기화를 자연스럽게 표현할 수 있어야 한다.

셋째, 사용자와의 실시간 상호 작용을 지원할 수 있어야 한다.

넷째, 하이퍼미디어 표현 및 처리 방법을 가지고 있어야 한다.

다섯째, 다양한 M/H 응용을 손쉽게 작성할 수 있도록 응용 프로그래머를 위한 프로그래밍 인터페이스를 지원해야 한다. 이러한 인터페이스는 시스템의 전체적인 구조나 데이터 표현 방식에 대한 세부적인 지식이 없어도 응용을 개발할 수 있어야 한다.

여섯째, 응용 프로그래머가 인터페이스를 사용하여 작성한 응용 프로그램에게 서비스를 제공해 주

는 플랫폼이 제공되어야 한다. 플랫폼은 응용 프로그램에서 API를 통해 요구하는 서비스를 제공하고, M/H 데이터를 표준 표현 방식으로 변환하여 네트워크나 데이터베이스에 전달하고, 표준 표현 방식으로 코딩된 M/H 데이터를 해석하여 응용 프로그램에 전달하는 기능을 수행한다.

위와 같은 요소 외에 표준을 사용하는 다른 시스템 사이에서 데이터 교환을 위한 네트워크/데이터베이스 인터페이스가 요구된다.

2.2 MHEG 표준

본 장에서는 다양한 멀티미디어 표준 중에서 실시간 대화형 멀티미디어 서비스를 목표로 설계된 MHEG 표준 및 MHEG을 기반으로 한 응용들을 분석한다. MHEG 표준은 분산 환경에서 이기종 플랫폼상의 멀티미디어 응용들 간에 최종 형태의 표현과 교환을 목적으로 멀티미디어 정보의 코드화된 표현을 정의한다. 이를 위해 MHEG 표준은 상호작용 및 멀티미디어 동기화, 실시간 표현, 실시간 상호교환, 최종형태의 표현 등을 그 범주로 두고 있으며, 특히 통신망을 통한 실시간 정보전송 및 교환에 그 초점을 맞추고 있다[16, 19, 20, 21, 22].

표준을 위해 객체 지향 방식의 접근을 수용하는 MHEG에서 멀티미디어 정보의 코드화된 표현을 MHEG 객체라 부르고, 객체 클래스는 일관성 있는 특정한 구조를 갖는 멀티미디어 객체들의 집합으로 정의하고 있다. 객체 지향적 접근 방식에 따라 MHEG 표준에서는 미디어 정보를 포함하고 있는 객체, 객체 사이의 관계, 객체의 동적인 행위, 객체의 실시간 조작을 최적화하기 위한 정보 등을 기술하기 위하여 클래스를 정의하고 있다[15]. 그림 1은 MHEG 객체 클래스의 상속도이다.

MH 객체 클래스는 가장 상위에 있는 클래스로서 모든 서브 클래스가 공통적으로 갖는 애트리뷰트들을 갖으며, 액션 클래스는 MHEG 객체와 런타임 객체, 그리고 채널의 초기 행위를 결정하는데 사용하는 클래스로서 객체 내의 코드 실행을 활성화시킨다. 이러한 액션 클래스의 인스턴스인 액션 객체들은 링크 효과를 기술하기 위하여 링크 객체 내에서 사용된다. 링크 객체는 공간적, 시간적, 그리고 조건 관계와 MHEG 객체, 런타임 객체, 채널에 대한 연결을 나타내기 위해서 정의된다. 링크 객체는 하나의 출발점 객체로부터 하나 이상의 목적지 객체들 사이의 관계를 나타내는 기능을 가지며, 링크의 개시를 위한 트리거 조건과 부수적인 조건, 그리고 조건이 만족됐을 때의 링크 효과를 나타내는 애트리뷰트를 가지고 있다.

그림 1 MHEG 객체 클래스 상속도

스크립트 클래스는 외부 함수 또는 프로그램에 대한 인터페이스와 관련한 클래스이다. MHEG 표준에서는 스크립트 언어에 대한 정의를 기술하고 있지 않지만, 이 클래스를 이용하여 응용이 사용하거나 지원할 수 있는 스크립트 언어의 유형 및 스크립트 디코더에 대한 정보를 표현할 수 있는 구조를 제공하고 있다. 내용 클래스는 내용 프리젠테이션을 위해 요구되는 정보를 포함하는 파라미터의 집합을 가지며, 미디어 정보의 코드화된 표현을 포함하거나 참조하기 위한 클래스이다. 미디어 데이터는 국제 표준에 따라 부호화된다. 다중 내용 클래스는 내용 클래스의 서브 클래스로 여러 가지 멀티미디어 데이터의 코드화된 표현을 포함하거나 참조하는 것과 관련한 애트리뷰트를 정의한다. 복합 클래스는 멀티미디어/하이퍼미디어 정보의 교환을 위한 구조와 객체들 간에 동기화를 표현하는 구조를 갖는 클래스이다. 크기, 속도, 볼륨, 선택 스타일에 대한 내용의 프리젠테이션과 다양한 표현 대상 사이의 시공간적인 관계와 특정 조건에 따라 수행되는 링크, 그리고 링크에 의해 수행되는 액션에 대한 반응 행위를 기술하는 애트리뷰트를 가진다. 복합 객체는 내용 객체와 복합 객체를 포함할 수 있으며, 조건에 따라 객체에 대한 행위를 발생시킬 수 있는 링크 객체, 링크 효과에 의해 시공간적인 동기화 표현과 관련된 액션 객체를 포함할 수 있다. 이것은 화면 단위의 프리젠테이션을 지원해야 하는 경우 기본 단위로 사용할 수 있는 구조이며, 전송시 여러 객체를 묶는 컨테이너로 사용가능하다. 컨테이너 클래스는 다중 객체들을 하나

의 집합으로 구성하여 교환하기 위한 패키지를 제공하는 클래스이다. 컨테이너 클래스는 객체가 준비됐을 때 수행되는 링크와 해제됐을 때 수행되는 링크를 가지며, 집합으로 포함하는 객체들의 인덱스와 종류를 기술할 수 있는 구조를 지원한다. 서술 클래스는 표현에 대한 객체들의 정보를 인코드하기 위하여 사용되며, 서버 혹은 다른 응용들이 실시간 표현에서의 객체 전송과 관리를 최적화하기 위한 정보의 요약과 관련된 클래스이다. 서술 클래스는 MHEG 객체를 사용하는 응용들의 초기화, 동작 수행, 관리를 처리할 의도로 사용되며, 관련 객체들의 집합, 객체 정보, 채널 정보, 상호 작용 스타일 정보 등을 기술할 수 있는 구조를 지원한다. 컴포넌트 및 모델 클래스는 상속받는 애트리뷰트외에 새로이 추가되는 애트리뷰트를 갖지 않으며 서브클래스를 묶어주는 역할을 하는 클래스이다.

3. MediaADE의 설계 및 구현

본 장에서는 MHEG을 기반으로 하는 멀티미디어/하이퍼미디어 응용 개발 환경(MediaADE :MHEG-based multimedia/hypermedia Application Development Environment)을 설계 및 구현한다.

3.1 MediaADE 설계

MediaADE는 계층적 구조로서 각 계층간에는 인터페이스가 존재하여 계층간 독립성 및 이식성을 극대화할 수 있는 개방형 구조이다. 각 계층은 실행, 추상화, 저장 계층으로 구성되며, 실행 계층과 추상화 계층간에는 프로그래밍 인터페이스가, 추상화 계층과 저장 계층간에는 네트워크/데이터베이스 인터페이스가 존재한다. MediaADE는 정보 표현 형식으로서 MHEG을 사용하는 개방형 M/H 응용 개발 플랫폼이다. 추상화 계층은 M/H 정보를 해석하기 위한 MHEG 엔진으로서 MHEG 객체를 효율적으로 해석할 수 있는 객체 관리 및 시공간 동기화 처리 구조와 인코딩/디코딩 및 링크/액션 처리 기능을 포함하고 있다. 프로그래밍 인터페이스는 MHEG 객체를 조작할 수 있는 DLL 형태의 응용 프로그래밍 인터페이스이며, 네트워크/데이터베이스 인터페이스는 ODBC와 표준 통신 프로토콜을 지원한다. 따라서, 실행 계층은 응용 개발자가 MediaADE에서 제공하는 API를 이용하여 개발한 응용이 되며, 저장 계층은 개발 환경에 따라 ODBC를 지원하는 데이터베이스 관리 시스템이 된다.

MediaADE는 클라이언트/서버 환경으로서 클라이언트에 MHEG 엔진과 API를 두고, 서버에 데이터베이스를 두어, MHEG 객체의 시공간 동기화 정보, 객체 상호 관계 등을 클라이언트에서 해석하여 클라이언트의 계산 능력을 최대로 활용하고 서버의 부담을 최소화할 수 있다. 또한 DBMS와 데이터베이스를 처리하는 응용만을 서버에 두어 클라이언트와 서버 사이의 통신량을 최소화한다. MHEG 엔진 설계는 각 M/H 객체의 속성을 객체 지향적인 방식으로 모델링 함으로써 저장과 전송시 뿐만 아니라 개발과 프리젠테이션 시에도 동일한 형태의 객체를 사용하여 모델링할 수 있다. 네트워크/데이터베이스 인터페이스는 MHEG 객체를 데이타베이스 관리 시스템에 저장/검색하는 기능을 제공한다. 응용 개발을 위한 인터페이스로서 개발자와 사용자가 직관적으로 이해할 수 있는 객체지향형 API를 정의한다.

MediaADE는 분산 환경에서 구동되는 개발 환경으로서, 그림 2는 전반적인 구조를 나타내고 있다.

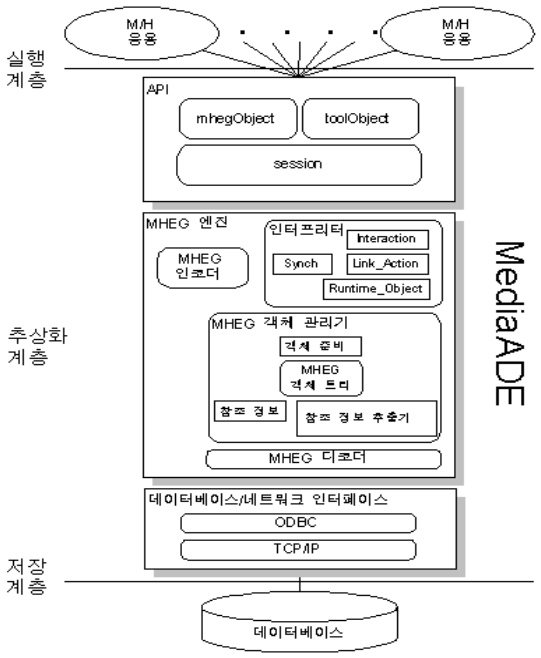

그림 2 MediaADE의 구조

그림 2에서 저장 계층은 MHEG 객체와 미디어 화일을 위한 데이터베이스, 통신상에서의 미디어 화일 및 MHEG 객체 전송과 데이터베이스에의 자료 저장을 위한 서버로 구성된다. 저장 계층은 실제 물리적인 멀티미디어 데이터와 MHEG 객체를

저장하는 데이터베이스와 이를 위한 인터페이스, 그리고 네트워크 상에서의 자료 전송을 위한 네트워크 인터페이스로 구성된다. 데이터베이스는 기존의 관계형 데이터베이스를 사용하게 되며, DBMS 종류와 관계없이 동일한 인터페이스를 갖도록 하며, 네트워크는 표준으로 정립되고 있는 TCP/IP를 사용하여 구현한다.

3.2 MHEG 엔진

분산 환경에서의 M/H 응용을 위한 MHEG 엔진은 MHEG 객체의 병행 처리가 가능한 선점형(preemption) 운영체제 기반에서 개발되어야 하며, MHEG 객체의 생성 기능 및 해석을 위한 내부 형식으로의 변환 기능, 객체의 프리젠테이션을 효율적으로 수행하기 위한 처리 기능을 가져야 한다. MediaADE에서 사용할 MHEG엔진의 구체적인 서술은 참고문헌[23]에 있다. 그림 3은 설계한 MHEG 엔진의 구성 모듈과 객체의 흐름을 나타낸 전체 구성도이다. MHEG 엔진은 네개의 모듈, 즉, MHEG 인코더, MHEG 디코더, MHEG 객체 관리기, 인터프리터 모듈로 구성된다.

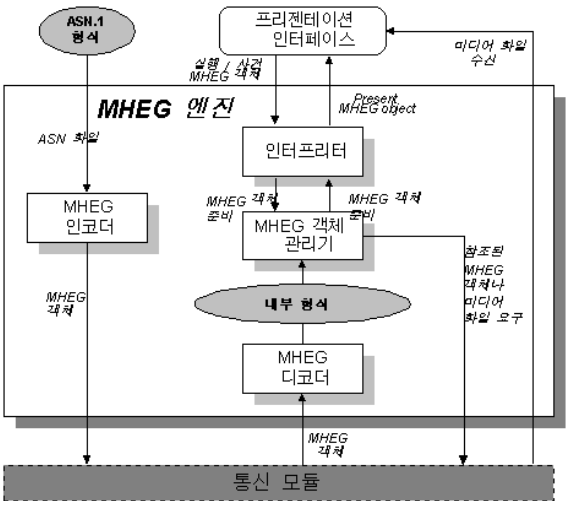

그림 3 MHEG 엔진 구성도

MHEG 인코더는 ASN 화일을 입력으로 받아 ASN.1(Abstract Syntax Notation One) 인코딩 규칙에 의해 MHEG 객체를 생성하는 역할을 하고, MHEG 디코더는 다양한 매체를 통해 로드된 MHEG 객체의 디코딩 기능 및 내부 형식으로의 변환 기능을 수행한다. MHEG 객체 관리기(object handler)는 내부 형식으로 변환된 MHEG 객체에 대한 관리 및 프리젠테이션을 위한 객체를 준비하는 역할을 한다. 객체에 대한 준비는 인터프리터로부터 준비(Prepare MHEG object) 메시지가 들어올 때 시작되며, 준비 메시지는 준비할 MHEG 객체의 식별자에 대한 포인터를 포함한다. 객체에 대한 준비 작업은 참조를 요구하는 MHEG 객체나 미디어 화일들에 대한 로드이며, 로드가 되어 MHEG 객체 관리기에 등록됐을 때, 준비가 끝났음을 알리는 메시지(Prepared MHEG object)를 인터프리터로 보낸다. 이 메시지 또한 준비된 MHEG 객체의 식별자에 대한 포인터를 포함한다. 인터프리터는 MHEG 객체 관리기를 통하여 준비된 객체로의 접근, 해석 및 동기화된 표현 기능 그리고 사용자로부터의 이벤트를 받아 처리하는 기능을 수행한다.

MHEG 표준에서는 MHEG 객체를 표현하기 위한 형식으로 ASN.1, SGML, 그 외의 방법 등을 기술하고 있으며, 기본적인 방식으로는 ASN.1(Abstract Syntax Notation.1 (ISO 8824) 표기법[24]을 사용하고 있다. 따라서, MediaADE에서는 MHEG 객체 표현 형식으로 ASN.1 표기법을 선택하였다. ASN.1 표기법은 MHEG 표준에서 모든 데이터 구조를 형식화하여 표현하는 수단이다. ASN.1 표기법으로 생성된 MHEG 객체는 전송을 위한 문법인 ASN.1 인코딩 규칙(Basic Encoding Rules for ASN.1 (ISO 8825) [25]에 의해 전송 형태로 변환된다.

3.3 응용 프로그래밍 인터페이스(API)

본 절에서는 API를 객체 지향적인 방법을 통해 설계한다. 사용되는 모든 객체들은 클래스로 구현되며, 각 객체들은 자신들만의 생성, 소멸, 준비, 실행 등의 메소드(method)를 갖는다. 생성되는 객체는 MHEG 엔진을 통해 해석되어 프리젠테이션시 자신의 실행 메소드를 사용하여 사용자에게 보여지게 된다. 이러한 객체들의 생성 및 소멸등을 지원하기 위하여 객체들을 조작할 수 있는 API를 DLL 형태로 제공한다. 따라서 응용 프로그래머는 제공되는 API만으로 MHEG 표준 기반의 응용 프로그램을 작성할 수 있다. 또한, DAVE나 MuX와는 달리 장치에 종속적인 클래스는 API 내부에 은폐시킴으로써 응용 프로그래머가 객체의 실제 전송 과정이나 프리젠테이션 과정들에 관계없이 응용 프로그램을 작성할 수 있도록 한다. 이는 장치 종속적인 미디어들의 세부적인 프리젠테이션과 저작에는 단점이 될 수 있으나, MHEG 표준은 이러한 미디어들의 표준 형태를 제시하고 있으므로 Windows NT환경에서는 동종의 디바이스 드라이버를 설치함으로써,

장치들의 특성에 적합하면서도 동일하게 프리젠테이션되는 미디어들을 사용할 수 있다.

가. API 클래스

응용 프로그래밍의 효율을 위한 API를 제공하기 위해 M/H 객체들의 클래스를 정의하고 해당 클래스에 속성들과 메소드들을 정의하여 미디어 객체들을 출력시킬 수 있도록 구성한다. 이는 객체 지향적인 설계로 가능하다. 객체 지향적인 방법을 사용하여 분산 M/H 응용 개발 환경을 위한 프로그래밍 인터페이스를 설계한다. 다양한 분산 M/H 응용을 개발하기 위한 프로그래밍 인터페이스는 카메라나 마이크 등의 미디어 장치들을 소프트웨어 자원으로 추상화하여 개발자가 장치 종속적인 부분에 관계없이 응용 프로그램의 개발이 가능해야 하며, 이러한 장치에 종속적인 모듈들의 확장에 대해서도 쉽게 모듈의 추가가 가능하여 강력한 이식성을 보장할 수 있다.

API는 개발자에게 단순하고 강력한 프로그래밍 패러다임을 제공할 수 있으며, 이는 개발자가 단순히 API에서 정의한 몇 개의 함수를 호출함으로서 분산 M/H 응용 프로그램을 개발할 수 있다는 것이다. 그림 4는 API 클래스의 상속도이다.

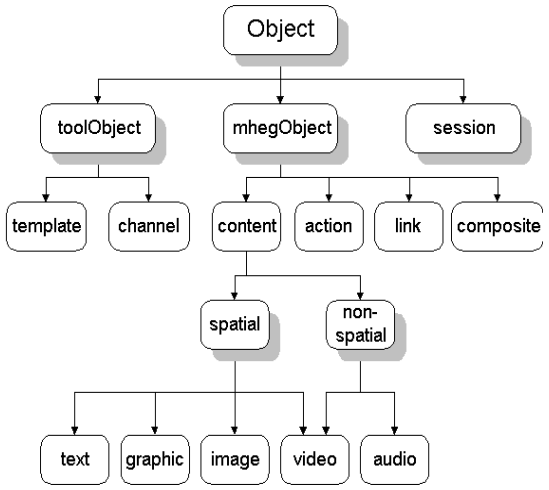

그림 4 API 클래스 상속도

M/H 응용들은 일반적으로 다양한 물리적 장치들을 제어할 수 있는 기능을 가져야 한다. API는 소프트웨어 추상화 기법을 사용하여, 물리적인 장치들을 가능한 한 사용자로부터 은폐시키는 방법을 사용한다. 이러한 API는 단지 간단한 프로그래밍 인터페이스만을 제공하는 것이 아니라 소프트웨어의 재사용에 대한 기능들도 제공한다. API의 하부 구조의 모듈들이 변경되더라도 API에서 정의한 기능들을 준수한다면 API를 사용하여 개발된 응용 프로그램은 쉽게 이식할 수 있게 된다.

API는 현재 표준으로 자리잡은 MPEG, JPEG 등의 동적 미디어에 관련된 루틴들이 존재하여 다양한 응용을 개발할 수 있다고 해도, 이후 새로운 표준을 추가할 수 있어야 한다. 새로운 미디어 형태의 지원을 위해서 제안하는 모델 내의 API에서는 플러그-인(plug-in)개념을 도입한다. 각종 하드웨어와 미디어 형태에 따른 루틴들을 집적시키고 추상화하고, 전송된 미디어의 종류를 판별하여 그에 따른 루틴들을 실행시키는 형태이다.

최상위의 Object 클래스는 다른 모든 클래스의 상위 클래스로서 하위 클래스에 상속될 속성들을 정의하는 역할만을 담당하는 클래스이다. toolObject 클래스는 응용 프로그램 개발시 사용되는 채널과 템플리트에서 사용될 속성들이 선언되어 있다. channel과 template 클래스는 응용 프로그램 개발시 저작자에게 도움을 줄 수 있는 개념으로, 템플리트는 하이퍼미디어 문서의 구조를 정의하는 개념으로 이후에 재사용될 수 있도록 하여 응용 프로그램 개발의 편의를 돕도록 하는 클래스이며, 채널은 미디어들을 물리적인 장치로 매핑시켜 주는 기능을 수행하는 클래스이다.

mhegObject클래스는 응용 프로그래밍 그리고 저장과 전송에 관계되는 모든 클래스들의 상위 클래스이다. mhegObject 클래스에는 생성과 소멸, 준비와 실행, 그리고 전송을 위한 인코딩과 디코딩을 위한 메소드가 선언되어 있다. mhegObject 클래스에서는 MHEG 객체들 중에 응용 프로그램 개발시에 사용하는 클래스들이 상속된다. content 클래스는 실제 화면과 스피커를 통해 출력되는 미디어 객체들을 총칭하는 클래스이다. 객체의 생성과 소멸, 준비와 실행, 중단을 위한 메소드들이 정의된다. action 클래스는 링크 연산의 결과로 발생하는 모든 행위들을 정의하는 클래스이다. 액션들은 MHEG 엔진에서 분석하여 실행시킬 수 있도록 메시지의 형태로 MHEG 엔진에 전송된다. 메시지 전송과 객체의 생성, 소멸에 대한 메소드들이 정의된다. link 클래스는 사용자가 링크 연산을 위해 준비된 객체를 선택했을 때 실행되는 클래스로써 생성과 소멸, 준비와 실행을 위한 메소드들이 정의된다. composite 클래스는 하나의 프레임을 나타낼 수 있는

클래스로서, action과 link, 그리고 content 클래스들을 리스트의 형태로 관리한다. 리스트로 연결된 객체들의 분석은 MHEG 엔진에서 담당하며, 실제 composite 클래스 형태로 생성된 클래스에서는 각 객체들에게 실행을 위한 메시지를 전달함으로써 해당 객체들을 실행시킬 수 있다.

나. 세션을 이용한 통신

session 클래스는 MediaADE와 사용자를 연결시켜 주는 교량의 역할을 담당한다. session 클래스는 데이터베이스가 존재하는 호스트 이름과 접속한 사용자 정보, 그리고 세션에 관계되는 정보를 저장할 수 있는 자료 구조를 가지고 있다. 또한 데이터베이스 접속을 위한 메소드와 전송되는 이벤트들을 처리하기 위한 메소드들이 정의된다.

session 클래스를 생성하여 초기화함으로써 사용자는 데이터베이스에 접속할 수 있고, 사용자 이름과 비밀번호를 검증받으며 자격에 맞는 권한을 부여받게 된다. session 클래스는 특정 사용자가 세션을 열고, 종료한 시간과, 사용자들에 대한 자료를 액세스할 수 있는 메소드, 그리고 데이터베이스 서버에 접속하는 메소드와 사용자의 권한을 검증, 부여하는 메소드를 가진다. session 객체는 하나의 응용 프로그램에서 하나만 존재하며, 응용 프로그램의 시작과 함께 생성되었다가 응용 프로그램의 종료와 함께 소멸된다. 응용 프로그래머는 MediaADE를 사용하고자 할 때에 session 객체를 생성시켜 사용할 수 있으며, MediaADE의 사용이 끝났을 때 소멸시킬 수 있다.

프로그래머는 MediaADE에서 제공하는 API를 사용할 수 있으며, 세션을 통해서만 MediaADE와 통신할 수 있다. 세션 객체와 응용 프로그램간의 상호작용은 그림 5와 같다.

그림 5에서 응용 프로그램은 세션 API에 세션 객체 생성을 요구한다. 사용자의 요구를 접수한 API는 세션 객체를 생성시켜 응용 프로그램에 할당한다. 이후 사용자의 모든 요구는 세션 객체를 통해서만 이루어지며 세션 객체는 발생되는 모든 이벤트나 메시지를 처리하기 위한 메카니즘을 갖는다. 세션 클래스의 객체는 응용 프로그램과 하부의 MHEG 엔진에서 발생될 수 있는 모든 상황을 처리하기 위한 메소드를 갖는다. 또한 접속시 사용자의 인증 작업 및 MHEG엔진에서의 모든 자료 전송 요구 등을 담당한다. 세션 객체는 다음의 세가지 형태의 자료 처리 루틴을 갖는다.

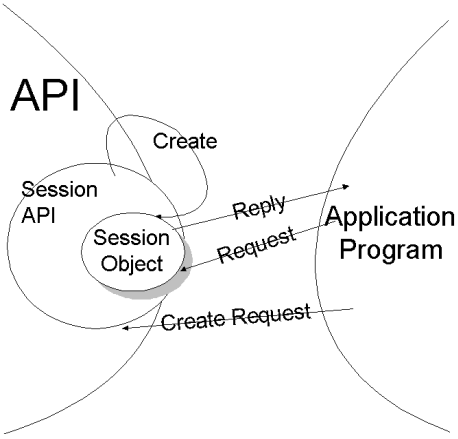

그림 5 세션의 역할

이벤트 처리 루틴은 발생하는 이벤트들을 해석하여 MHEG 객체이면, 응용 프로그램에서 처리할 수 있도록 전송하고, 일반적인 메시지이면 메시지 처리 루틴으로 전송한다. 응용 프로그램은 이벤트 처리 루틴을 사용할 때 특정한 이벤트 식별자를 주어 호출하게 되지만 전송되는 이벤트는 사자가 원하는 이벤트가 아닐 경우가 있으므로 응용 프로그램에서는 이를 예외 처리 루틴이나 메시지 처리 루틴으로 전송하는 부분이 존재해야 한다. 메시지 처리 루틴은 응용 프로그램에서 발생된 메시지를 해석하여 MHEG 엔진으로 전송하기 위해 이벤트 처리 루틴으로 전송하고, 엔진에서 이벤트 처리 루틴을 통해 전송된 메시지를 해석하여 사용자에게 적절한 메시지를 출력한다. 세션에서 존재할 수 있는 예외는 원하던 자료가 전송되지 않았을 경우나 실제 전송될 시간보다 늦게 전송되었을 경우이다. 예외 처리 루틴은 기본적으로 사용자에게 오류 메시지를 출력하고 종료하는 기능만을 가지고 있으나, 응용 프로그래머는 적절한 기능을 갖는 예외 처리 루틴을 오버로드(overload)하여 정의할 수 있다. 이벤트 처리 루틴은 발생된 모든 이벤트를 수집하여 분류하는 작업을 수행한다. 사용자가 요구하는 이벤트와 예외 발생을 제외한 모든 이벤트는 메시지 처리 루틴에서 처리된다. 메시지 처리 루틴에서 처리할 수 없는 형태의 이벤트는 예외 처리 루틴에서 처리한다. 응용 프로그램은 세션 클래스의 이벤트 처리 루틴에서 분류한 이벤트들을 자신의 프로세스 내에서 사용하거나, 예외 처리 루틴이나 메시지 처리 루틴에서 처리해 주도록 의뢰할 수 있다. 또한 세션 객체는 사용자의 접속에 필수

적인 식별자와 비밀 번호를 인증받기 위하여 서버로 전송하고, 사용자가 접속한 클라이언트에서 데이터베이스 서버까지의 가상 전송로를 생성한다.

다. API 함수

API는 API 클래스의 메소드로 정의된다. API는 크게 객체를 생성시키는 함수와, 준비시키는 함수, 실행시키는 함수, 그리고 소멸시키는 함수로 구분된다. 객체의 생성에 관련되는 함수는 C++ 언어의 생성자 함수를 사용한다. 미디어 객체가 생성되면, 객체는 자신의 미디어 화일의 핸들과 함께 초기화된다. 객체의 준비에 관련되는 함수는 생성된 객체에 해당되는 속성을 부여함으로써 이루어진다. 준비된 객체는 사용자가 직접 볼 수 있는 형태로 출력되기 위하여 채널에 할당되고, 실행 함수가 수행되기를 기다린다. 객체를 실행시키는 함수는 각 객체마다 수행 방식은 달라지게 된다. 텍스트나 이미지 등의 정적인 미디어들은 한번의 출력으로 실행이 종료되지만 음향, 음성, 동영상등은 실행되는 시간에 영향을 받는다. 미디어 형태에 따른 전역적인 속성들은 미리 채널에 정의되므로 각 미디어 객체들은 지속시간 등 자신이 프리젠테이션될 때의 속성만을 가지고 있다. 모든 객체들은 채널을 통해서 출력되므로 실행은 채널의 실행 함수를 호출하는 것이다. Content 클래스의 실행 함수는 자신에게 할당된 채널들의 실행 함수를 호출함으로써 프리젠테이션을 수행한다. execute() 함수는 mhegObject 클래스에서 상속받아 미디어 특성에 따른 출력 방식을 정의하고 있는 함수이다. composite 클래스 객체는 다른 클래스의 객체들을 하위에 포함하고 있으므로, execute() 함수의 실행은 깊이 우선 탐색 방식으로 하위 객체들을 실행시킨다. 최하위 객체들 중 content 클래스 객체들은 즉시 실행되고, link 클래스 객체는 Ready() 함수가 호출되어 사용자의 입력을 기다린다. 준비 상태에 있는 link 객체들을 선택하면 link 객체내에 포함되어 있는 action 클래스 객체들을 실행시키게 된다. 만일, composite 클래스 객체 내의 하위 클래스가 composite 클래스 객체인 경우 다음 객체들을 읽어들여 수행시키는 것을 중단하고 함수를 종료시키며, 하위의 composite 객체를 실행시키라는 메시지를 전송한다. 소멸 함수는 생성 함수와 마찬가지로 C++ 언어의 소멸자 함수를 사용한다. 특히, composite 클래스의 객체의 경우는 리스트 형태로 내포하고 있는 다른 클래스의 객체들을 소거시켜 주고 난 후에 소멸되어야 한다.

라. 응용 프로그래밍

응용 프로그래밍은 크게 준비 단계와 실행 단계, 그리고 종료 단계로 구분할 수 있다. 준비 단계는 세션 객체의 초기화와 각 객체들의 준비, 그리고 MHEG 엔진의 로딩으로 구성된다. 실행 단계는 해당 객체들의 전송과 실행으로 구성되며, 종료 단계는 세션의 종료와 세션 객체의 소멸로 구성된다. 세션의 초기화는 세션 객체를 생성하고 초기화하는 함수의 호출, 그리고 데이터베이스 서버로의 접속 함수로 이루어진다.

저작 응용 프로그램의 경우는 미디어 화일마다 하나의 객체를 생성해 내므로 각 객체들의 준비는 프리젠테이션 응용 프로그래밍 시에만 요구된다. MHEG 엔진에서 번역되어 전송되는 메시지들에 따라 필요한 객체를 생성하고, 준비한 후에 각 객체들의 소멸시키는 작업은 세션 객체가 MHEG 엔진에 요청하여 수행된다. 저작 응용 프로그래밍시에는 사용자의 요구에 따라 해당하는 객체들을 생성하고, 속성 정의 다이얼로그를 열어 사용자의 입력을 받아들이고, 사용자의 최종 결정에 의해 MHEG 객체로 생성된다. 이는 MHEG 엔진에서 언급했던 것처럼 인코딩되어 데이터베이스로 전송되게 된다. 사용했던 객체들의 소멸과 서버를 종료시키는 작업은 서버를 생성하고 초기화하는 작업의 역순으로 행해진다.

전체 하이퍼미디어 문서를 반영하는 composite 클래스의 객체는 또다른 여러개의 composite 클래스 객체를 포함할 수 있다. 최하위 단계의 composite 클래스 객체가 전송되면, 해당 노드의 프리젠테이션을 시작하게 된다. 예외 처리를 위해서는 session 클래스 객체에서 제공하는 메시지 처리기와 사건 처리기를 사용하여 응용 프로그래머가 처리해 주어야 한다. composite 클래스 객체의 실행 함수를 수행하고 난 후의 제어권은 MHEG 엔진으로 전송된다. MHEG 엔진 내의 인터프리터가 이후의 모든 프리젠테이션을 수행한다.

3.4 네트워크/데이터베이스 인터페이스

본 절에서는 데이터베이스에 MHEG 객체를 저장/검색하기 위한 데이터베이스 스키마를 정의하고, MediaADE의 네트워크/데이터베이스 인터페이스를 설계 및 구현한다.

가. MHEG 객체를 위한 데이터베이스 스키마

하이퍼미디어를 지원하기 위해 요구되는 MHEG 객체 클래스는 복합 클래스, 내용 클래스, 링크 클래스, 액션 클래스 등과 응용 프로그램에서 한 화면을 나타내기 위한 프레임(frame) 클래스가 있다. 프레임은 복합 객체로부터 상속된 클래스이다. 실제 데이터베이스에서 MHEG 엔진으로 데이터가 전송될 때 프레임이 전송 단위로 사용된다.

본 절에서는 MHEG의 각 객체 저장을 위한 데이터베이스 스키마를 정의한다. 그림 6은 전체적인 구조를 보이는 개체 관계 다이어그램(Entity-Relationship Diagram)이다.

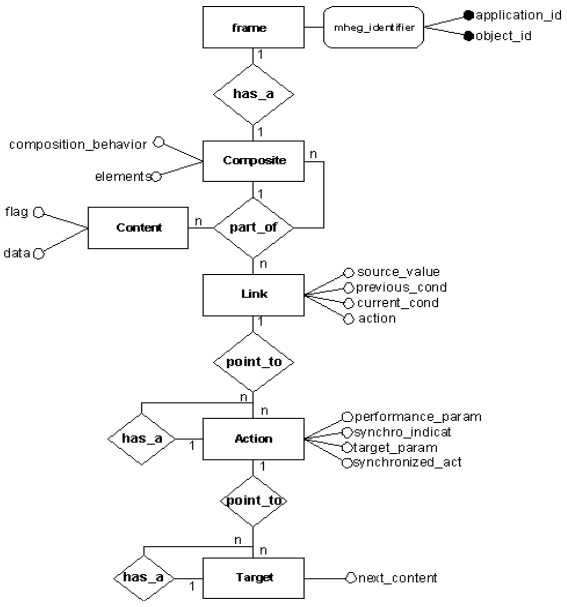

그림 6 MHEG 데이터베이스 스키마 ER 다이어그램

프레임 클래스(이하 frame)에는 이 객체에 포함된 모든 객체에 대한 정보, 표현 정보 등이 포함된다. mheg_identifier는 복합 애트리뷰트로서 application_id와 object_id를 가지고 있다. 이 두 필드의 조합이 frame을 구별하기 위한 기본 키(mheg_identifier)로 사용된다. application_id는 MHEG 객체를 저작한 시스템의 IP 주소 값으로 네트워크에 연결된 전세계 모든 시스템에서 유일한 값이 된다. 그리고 object_id는 각 시스템에서 객체 생성시 순차적으로 부여한 번호로써 각 시스템 내에서 유일한 값이 된다. frame 테이블에는 실제 포함된 데이터를 가지는 Composite 테이블을 가리키는 필드가 있다. 복합 클래스는 실제 동기화 정보 등을 포함하고 있는 하이퍼미디어 표현의 기본 단위가 된다. composition_behavior는 사전 정의 행위나 특정 행위를 나타내기 위해 사용된다. elements는 복합 객체 내에 포함된 요소들을 가리키기 위해 사용된다. 내용 클래스는 실제 데이터나 참조를 위한 정보를 갖는다. flag는 data 필드의 내용이 실제 데이터의 내용인지, 참조를 위한 경로 정보인지를 나타내기 위해 사용된다. 링크 클래스는 하이퍼텍스트의 링크 개념을 지원하기 위해 사용된다. 객체의 상태나 속성을 나타내는 source_value가 링크 조건 검사시 previous_cond 값을 가지고 있다가 current_cond으로 변하면 링크가 발생하게 된다. action은 링크 발생시 객체의 행위를 기술하는 action 클래스를 가리키기 위해 사용된다. performance_param은 액션의 반복 횟수를 나타낸다. synchro_indicat는 액션이 병행 수행인지 순차 수행인지를 나타내기 위해 사용된다. target_param은 액션 수행시 동시에 수행될 일련의 타겟을 가리키는 참조 값을 갖는다. synchoronized_act는 액션이 단일 액션인지 또 다른 액션을 포함하는 복합 액션인지를 나타내기 위해 사용된다.

그림 7은 클라이언트와 서버의 네트워크/데이터베이스 인터페이스 구성이다.

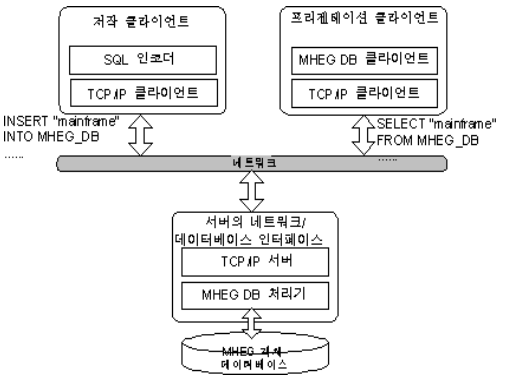

그림 7 클라이언트와 서버의 네트워크/데이터베이스 인터페이스

나. 클라이언트 인터페이스

클라이언트 인터페이스는 MHEG 객체 저장을 위해 SQL 삽입 문장을 생성하는 SQL 인코더, 서버와 네트워크를 통해 데이터를 주고받는 TCP/IP 클라이언트, MHEG 객체를 엔진에 전달하기 위해 데이터베이스에서 MHEG 객체를 검색하는 MHEG DB 클라이언트가 있다.저작 클라이언트에서 작성된 MHEG 객체는 네트워크/데이터베이스 인터페이스에 의해 전송 및 저장된다. SQL 인코더와 TCP/IP 클라이언트가 네트워크/데이터베이스 인터페이스를

구성한다.

SQL 인코더는 MHEG 객체를 분석하여 이를 데이터베이스에 저장하기 위한 SQL 문장을 생성한다. 또한 ODBC 데이터베이스 인터페이스[26]를 통해 DBMS에 질의 처리를 요구할 수 있다. 알고리즘 1은 SQL 인코더에서 SQL문장을 생성하는 알고리즘이다.

알고리즘 1 SQL 문장 생성을 위한 알고리즘

```
SQL_Encoder(MHEG *mheg_obj)
// mheg_obj : 저작 객체를 MHEG에 의해 코딩한 MHEG객체
begin
  int  app_id, obj_id;
  DStream *pDS;
  BOOL constructed = FALSE;
  Long *lpSQL;
  constructed = Get_MHEG_ID(mheg_obj, app_id, obj_id);
  lpSQL="INSERT INTO"+frame+" VALUES"+mheg_obj+";";
// frame: 화면 단위의 MHEG객체 저장을 위한 최상위 테이블
  if(constructed)
    begin
     refd = GetRefereced(mheg_obj);
     pDS=lpSQL+"INSERT INTO"+refd+"VALUES"+refd+";";
    end
  else
    begin
     pDS = lpSQL;
    end
  Call TCP_IP_Client(pDS, SAVE_MHEG);
end
```

TCP/IP 클라이언트는 SQL 인코더가 생성한 바이트 스트림을 네트워크를 통해 데이터베이스 서버에 전달하는 역할을 담당한다. TCP/IP 클라이언트는 저작, 프리젠테이션 시에 발생하는 서로 다른 요구를 받아들이기 위해 두 가지 부분으로 분리되고, TCP/IP 클라이언트 호출시 함께 전달되는 메시지에 의해 해당 부분이 수행된다. 이때 전달되는 메시지는 저작 클라이언트에서 발생하는 "SAVE_MHEG" 메시지와 프리젠테이션 클라이언트에서 발생하는 "SELECT_MHEG" 메시지가 있다.

알고리즘 2는 MHEG 객체 저장, 검색을 위해 서버와 데이터를 교환하는 통신 모듈 알고리즘이다.

프리젠테이션 클라이언트의 네트워크/데이터베이스 인터페이스를 통해 MHEG 객체가 서버로 부터 전달된다. MHEG DB 클라이언트와 TCP/IP 클라이언트가 네트워크/데이터베이스 인터페이스를 이루는데, TCP/IP 클라이언트는 저작 클라이언트에서와 같다.

알고리즘 2 MHEG 객체 저장, 검색을 위한 통신 모듈 알고리즘

```
TCP_IP_Client(DStream *pDS, Message msg)
// pDS : 전송할 데이터스트림을 가리키는 포인터
// msg : MHEG 객체저장 질의문인지 검색 질의문인지 구별
begin
  pDataSocket = socket(); // 서버와의 통신을 위한 소켓 생성
  connect();              // 서버에 접속을 요구한다.
  switch(msg)
   begin
    case SAVE_MHEG:
     begin
      pDataSocket.send(pDS);
      pDataSocket.close();
      return;
     end
    case SELECT_MHEG:
     begin
      pDataSocket.send(pDS);
      pDS = pDataSocket.receive();
      pDataSocket.close();
      return pDS;
     end
   end
end
```

MHEG DB 클라이언트는 네트워크를 통해 전송받은 MHEG 객체를 MHEG 엔진에 전달한다. 이때 객체 관리기가 참조된 MHEG 객체에 대한 전송을 요구하면, 객체 검색을 위한 SQL 검색문을 생성하여 TCP/IP 클라이언트를 통해 데이터베이스 서버에 요구한다. 질의의 결과로 MHEG 객체가 전송되면 MHEG 엔진에 전달한다. SQL 검색문을 생성하는 것은 저작 클라이언트의 SQL 인코더에서와 같이 MHEG 객체 식별자를 기본 키로 사용한다.

프리젠테이션 클라이언트의 TCP/IP 클라이언트는 MHEG 객체를 데이터베이스 서버로부터 전송받아 MHEG DB 클라이언트에 전달하는 기능과 MHEG 엔진에서 요구하는 객체 검색을 위해 전송된 SQL 검색 문장을 데이터베이스 서버에 전달하는 기능을 수행한다.

다. 서버 인터페이스

서버 인터페이스는 TCP/IP 서버, MHEG DB 처리기로 구성된다. TCP/IP 서버는 저작 및 프리젠테이션 클라이언트로부터 접속 요구에 따라 통신을 위한 스트림 소켓을 생성한다. TCP/IP 서버는 클라이언트에서의 접속을 받아들이는 하나의 마스터와 각 클라이언트의 요구를 처리하고 결과를 반환하는 하나 이상의 슬레이브로 구성되어있다. TCP/IP 서버 마스터는 클라이언트로부터의 접속 요구를 검사하고 있다가 클라이언트가 접속을 요구하면

이를 받아들이고 접속된 클라이언트에 슬레이브를 할당한다.

알고리즘 3 MHEG 객체 저장, 검색을 위한 알고리즘

```
MHEG *MDB_Manager(Long *lpSQL)
//lpSQL: 데이터베이스 연산을 위한 SQL문장의 포인터
begin
  CDatabase m_MHEGdb;
  MHEG *mheg_obj;
  m_MHEGdb.Open(MHEGdb);
  //MHEG 객체를 위한 데이터베이스 테이블을 오픈한다.
  if(lpSQL == Insert Command)
    begin
     if(m_MHEGdb.ExecuteSQL(lpSQL) == OK);
//ExecuteSQL() 함수는 인자로 전달된 SQL 문장을 수행하고
//삽입 명령 처리 성공시 'ACK' 신호를 클라이언트에 전송
       call TCP_IP_Server(ACK, SendCtl);
     else
//삽입 명령 실패시 'NOACK' 신호를 보낸다.
       call TCP_IP_Server(NOACK, SendCtl);
    end
  else
    begin
     pDS = m_MHEGdb.ExecuteSQL(lpSQL);
     call TCP_IP_Server(pDS, SendMHEG);
    end
end
```

MHEG DB 처리기는 ODBC 인터페이스를 통해 DBMS에게 MHEG 객체의 삽입과 검색 SQL 문장을 전달하고, 관리기는 이 SQL 문장을 ODBC 함수를 호출함으로써 처리한다. ODBC 응용 프로그램은 응용 프로그램, ODBC 인터페이스, ODBC 드라이버의 3가지로 구성되어 있다. 응용 프로그램(본 논문에서는 MHEG DB 처리기)이 함수 형태로 제공되는 ODBC 인터페이스를 호출하여 질의 처리를 요구하면, ODBC DLL(ODBC 드라이버)은 DBMS가 처리할 수 있는 SQL 질의문으로 바꿔서 DBMS에 전달하여 질의 처리를 하고 결과를 응용 프로그램에 반환한다. 알고리즘 3은 MHEG DB 처리기에서 MHEG 객체를 저장, 검색하는 과정을 보여준다.

4. 결 론

분산 M/H 응용을 체계적으로 개발하기 위한 환경을 이용하여 개발하는 것이 생산성을 극대화할 수 있는 방법일 것이다. 따라서 분산 M/H 응용 개발 환경에 대한 연구는 매우 중요하며 파급 효과가 클 것이다.

따라서 본 논문에서는 멀티미디어/하이퍼미디어 응용 개발 환경인 MediaADE를 설계 및 구현하였다. MediaADE는 개방형 멀티미디어/하이퍼미디어 응용 개발 플랫폼으로서, 세개의 계층적 구조로 구성되어 있으며 계층간 독립성을 최대한 유지할 수 있는 구조이다. 개방형으로서 M/H정보 표현을 MHEG을 사용하였고, 네트워크와 데이터베이스에 독립적일 수 있도록 표준을 따르고 있다. 추상화 계층은 MHEG엔진으로서 MHEG 객체를 효율적으로 관리할 수 있는 객체 처리 및 시공간 동기화 처리 구조와 인코딩/디코딩, 상호 작용을 위한 액션/링크 등을 포함하였다. 프로그래밍 인터페이스는 MHEG 객체를 조작할 수 있는 동적 링킹 라이브러리 형태의 API로서 MHEG 객체 처리와 도구 및 통신을 위한 세션 등이 있다. 네트워크/데이터베이스 인터페이스는 ODBC와 표준 통신 프로토콜인 TCP/IP를 지원한다. 따라서, 실행 계층은 응용 개발자가 MediaADE에서 제공하는 API를 이용하여 개발한 응용이 되며, 저장 계층은 개발 환경에 따라 ODBC를 지원하는 데이터베이스 관리 시스템이 된다.

MediaADE를 이용한 응용으로서 원격 교육[27], 멀티미디어/하이퍼미디어 정보를 저작하고 프리젠테이션할 수 있는 저작 시스템[28, 29]등이 개발되고 있으며, MediaADE의 각 기능이 계속 보강되고 있다.

본 논문에서 제안한 M/H 응용 개발 환경인 MediaADE는 다음과 같은 장점들을 갖는다.

첫째, 계층적인 구조로써 계층간 인터페이스만의 변경으로 이기종 간의 접속이 용이하다.

둘째, 이질적인 분산 환경에서 응용들 간에 M/H 정보를 상호 교환할 수 있는 국제적 표준인 MHEG을 기반으로 하고 있으며, 이러한 MHEG 정보를 인코딩/디코딩 및 해석할 수 있는 MHEG 엔진을 내장하고 있다.

셋째, M/H 응용 개발을 위한 인터페이스는 객체 지향적 API 형태로서 응용 프로그램 개발자와 사용자 모두에게 직관적인 방법으로 멀티미디어 객체를 사용할 수 있게 해준다. 또한, M/H 데이터와 각 장치들의 세부적인 면을 개발자와 사용자들에게 은폐시킬 수 있다.

넷째, 객체 지향적 설계를 통한 하이퍼미디어 객체 및 유용한 클래스의 수준 높은 재사용이 가능하다.

본 논문의 결과는 초고속 정보 통신망상에서 분산 M/H 응용 개발을 위한 플랫폼으로 활용할 수 있을 것이다.

향후 연구되어야 할 부분은 클라이언트가 서버

의 주소에 투명하게 접근할 수 있게 하여 완전한 분산 환경을 지원하도록 하는 것이다. 이때 지역 서버가 네트워크에 산재되어 있는 모든 서버 접근을 위한 정보를 제공하도록 하여 클라이언트가 자신이 사용하는 지역 서버에 대한 접근 경로만을 알고 있으면 전체 서버들에 대한 접근을 할 수 있도록 한다. 따라서, 네트워크 상에 산재된 다양한 멀티미디어/하이퍼미디어 자원을 효율적으로 활용할 수 있는 분산 멀티미디어/하이퍼미디어 응용 개발의 용이성을 높일 수 있을 것이다.

참 고 문 헌

[1] R. Steinmetz and K. Nahrstedt, Multimedia: Computing, Communications & Applications, Prentice Hall Inc., 1995.

[2] M. Muhlhauser and J. Schaper, "Project NESTOR: New Approaches to Cooperative Multimedia Authoring / Learning," 4th International Conference on Computer and Learning, ICCAL'92, Springer-Verlag, pp.453-465, 1992.

[3] R. Kushwaha and J. Whitescarver, "Integration of Virtual Classroom and Multimedia on the Information Superhighway," International Conference Distributed Multimedia System and Applications, IASTED/ISMM, pp.135-138, 1994.

[4] J. M. Velez and M. R. Gomes, "DEDICATED - MODULAR TRAINING SYSTEM DELTA Project D2014," Proceedings of the Second International Workshop on Multimedia: Advanced Teleservices and High-Speed Communication Architectures, IWACA'94, (Ed.) R. Steinmetz, Springer-Verlag, pp300-314, 1994.

[5] J. M. Haake and B. Wilson, "Supporting Collaborative Writing of Hyperdocuments in SEPIA," Conference on Computer-Supported Cooperative Work, CSCW '92, ACM SIGCHI & SIGOIS, pp. 138 - 146, 1992.

[6] N. A. Streitz, "Putting Object to Work: Hypermedia as the Subject Matter and the Medium for Computer-Supported Cooperative Work," The Eighth European Conference on Object-Oriented Programming, ECOOP'94, (Eds.) M.Tokoro and R.Pareschi, Springer-Verlag, pp.183-193, 1994.

[7] J. F. K. Buford, Multimedia Systems, ACM Press, 1994.

[8] F. M. Kappe, Aspects of a Modern Multi-Media Information System: Dissertation of for the Academic Degree Doctor of Technical Sciences, Graz University of Technology, Australia, 1991.

[9] W. I. Grosky, "Multimedia Information Systems," IEEE Multimedia, Vol.1, No.1, pp.12-24, 1994.

[10] C. Bertin, "Eurescom IMS1 Projects(Integrated Multimedia Services at about 1 Mbit/s)," Proceedings of the Second International Workshop on Multimedia: Advanced Teleservices and High-Speed Communication Architectures, IWACA'94, (Ed.) R. Steinmetz, Springer-Verlag, pp.53-66, 1994.

[11] B. Furht, "Multimedia Systems : An Overview," IEEE Multimedia, Vol.1, No.1, pp.47-59, 1994.

[12] R. F. Mines, J. A. Friesen and C. L. Yang, "DAVE : A Plug and Play Model for Distributed Multimedia Application Development," Proceedings Second ACM International Conference on Multimedia, ACM Multimedia'94, pp.59-66, 1994.

[13] 한국전자통신연구소 분산멀티미디어 연구실, '95 MuX 사용자 그룹 워크샵 발표자료집, 1995.

[14] I. Herman, G.J.Reynolds, and J.Davy, "MADE: A Multimedia Application Development Environment," Proceeding of International Conference on Multimedia Computing and System, IEEE Computer Society Press, 1994.

[15] ISO/IEC DIS 13522-1 Information technology - Coding of Multimedia and Hypermedia information - Part 1: MHEG object representation - Base notation(ASN.1), 1994.

[16] J. F. K. Buford and C. B. Gopal, "Standardizing a Multimedia Interchange Format: A Comparison of OMFI and MHEG," Proceedings of the International Conference on Multimedia Computing and Systems, IEEE Computer Society Press, pp.463-472, 1994.

[17] M. B. Thomas and E. Wolfgang, "MHEG: Explained," IEEE Multimedia, Vol. 2, No. 1, pp.26-38, 1995.

[18] J. J. Sung, M. Y. Huh, H. J. Kim and J. H. Hahm, "Hypermedia Information Retrieval System Using MHEG Coded Representation in a Networked Environment," Proceedings of the Second International Workshop on Multimedia: Advanced Teleservices and High-Speed Communication Architectures, IWACA'94, (Ed.) R. Steinmetz, Springer-Verlag, pp.67-77, 1994.

[19] A. Rizk, F. Malezieux and A. Leger, "Distributed Hypermedia Link Service on WAN: An Environment with MHEG on the ATM Network," Proceedings of the Eurographics Symposium, (eds.) W. Herzner and F. Kappe, Springer-Verlag, pp. 18 - 28, 1994.

[20] F. Kretz and F. Colaitis, "Standardizing Hypermedia Information Objects", A Guided Tour of Multimedia Systems and Applications, (Reprinted from IEEE Communications Magazine, pp. 60-70, 1992), (Eds.) B. Furht and M. Milenkovic, IEEE Computer Society Press, pp. 402-412, 1995.

[21] R. Price, "MHEG: An Introduction to the Future International Standard for Hypermedia Interchange,"Proceedings of the First ACM International Conference on Multimedia, ACM Multimedia'93, pp.121-128, 1993.

[22] F. Colaitis and F. Bertrand, "The MHEG Standard: Principles and Examples of Applications," Proceedings of the Eurographics Symposium, (eds.) W. Herzner and F. Kappe, Springer-Verlag, pp. 3 - 17, 1994.

[23] 이세훈, 황창종, "멀티미디어/하이퍼디어 응용 개발

환경을 위한 MHEG 엔진 설계," 한국 정보처리 학회 논문지, Vol.3, No.2, 1996.
[24] ISO/IEC IS 8824 Specification of Abstract Syntax Notation One (ASN.1), Second edition, 1990.
[25] ISO/IEC IS 8825 Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1), Second edition, 1990.
[26] C. Moffat, "Desinging Client-Server Applications for Enterprise Database Connectivity", http://www.microsoft.com/MSDN/OFFRAMP/FASTRACK/TECHNOTE/HETDB.HTM
[27] 이세훈, 황창종외, "초고속 정보 통신망에서 원격 교육을 위한 개방형 시스템," 한국 정보과학회 가을 학술발표논문집, Vol.22, No.1, 1995.
[28] 인하대학교 부설 컴퓨터과학응용연구소(황창종), C 언어 교육용 소프트웨어 개발, 최종연구보고서, 미원정보기술(주), 1995.
[29] 인하대학교 부설 컴퓨터과학응용연구소(황창종), 분산 환경에서의 교육용 소프트웨어 개발을 위한 공동 저작시스템, 정보통신연구관리단, 1995.