분산 객체 컴퓨팅 환경에서 실시간 협약 및 적응 스트림 서비스를 위한 QoS 통합 플랫폼의 구축

전 병 택[†] · 김 명 희[†] · 주 수 종[††]

요 약

최근 인터넷 기반의 분산 멀티미디어 환경에서 가장 성장하는 기술로는 스트림 서비스 기술과 분산 객체 기술을 꼽을 수 있다. 특히, 분산 객체 기술에 스트림 서비스 기술을 통합하려는 연구들이 진행되고 있다. 이 기술들은 다양한 스트림 서비스 관리 모델과 프로토콜의 연구에서 적용되고 있다. 그러나, 기존에 제시된 관리 모델들은 스트림 전송의 서비스 질(QoS)에 대한 지원이 미흡하다. 또한, 서비스 질에 관련된 기능들이 특정 응용 서비스의 부속 모듈로 개발됨에 따라, 확장이나 재사용을 지원할 수 없는 문제점을 나타내고 있다. 이를 해결하기 위해 본 논문에서는 분산 객체 기술을 적용하여 확장 및 재사용이 용이하고 스트림의 서비스의 질을 보장하는 QoS 통합 플랫폼을 제안했다. 제안된 플랫폼의 구조는 사용자 제어 모듈, QoS 관리 모듈 및 스트림 객체의 세가지 컴포넌트로 구성된다. 스트림 객체는 TCP/IP상에서 RTP 패킷을 송·수신 기능을 한다. 사용자 제어 모듈은 CORBA객체를 이용하여 스트림 객체들을 제어한다. QoS 관리 모듈은 사용자 제어 모듈간에 서비스 질을 유지하는 관리 기능을 한다. QoS 제어는 앞에서 언급한 컴포넌트간의 상호작용을 통해 자원 모니터링, 협약과 자원 적응과정이 이루어진다. QoS 통합 플랫폼의 구축을 위해 관련 모듈들을 독립적으로 구현하고, 이들이 CORBA환경에서 플랫폼 독립성, 상호운용성, 이식성을 갖도록 그들간에 인터페이스들을 IDL로 정의하였다. 제안된 플랫폼의 구현을 위해 Solaris 2.5/2.7에 호환되는 OrbixWeb 3.1c, 자바언어와 Java Media Framework API 2.0, Mini-SQL1.0.16 및 관련 이미지캡처보드 및 영상카메라를 사용하였다. 본 플랫폼의 기능검증을 위한 결과로서, 플랫폼 상에서 스트림 서비스가 진행되는 동안, 클라이언트와 서버의 GUI를 통해 위에서 기술한 모듈들의 수행결과와 QoS 제어 과정으로부터 얻어지는 수치적 데이터를 보였다.

The Construction of QoS Integration Platform for Real-time Negotiation and Adaptation Stream Service in Distributed Object Computing Environments

Byung-Taek Jun[†] · Myung-Hee Kim[†] · Su-Chong Joo[††]

ABSTRACT

Recently, in the distributed multimedia environments based on internet, as radical growing technologies, the most of researchers focus on both streaming technology and distributed object technology. Specially, the studies which are tried to integrate the streaming services on the distributed object technology have been progressing. These technologies are applied to various stream service managements and protocols. However, the stream service management models which are being proposed by the existing researches are insufficient for supporting the QoS of stream services. Besides, the existing models have the problems that cannot support the extensibility and the reusability, when the QoS-related functions are being developed as a sub-module which is suited on the specific-purpose application services. For solving these problems, in this paper, we suggested a QoS Integrated platform which can extend and reuse using the distributed object technologies, and guarantee the QoS of the stream services. A structure of platform we suggested consists of three components such as User Control Module(UCM), QoS Management Module(QoSM) and Stream Object. Stream Object has Send/Receive operations for transmitting the RTP packets over TCP/IP. User Control Module(UCM) controls Stream Objects via the CORBA service objects. QoS Management Module(QoSM) has the functions which maintain the QoS of stream service between the UCMs in client and server. As QoS control methodologies, procedures of resource monitoring, negotiation, and resource adaptation are executed via the interactions among these components mentioned above. For constructing this QoS integrated platform, we first implemented the modules mentioned above independently, and then, used IDL for defining interfaces among these modules so that can support platform independence, interoperability and portability based on CORBA. This platform is constructed using OrbixWeb 3.1c following CORBA specification on Solaris 2.5/2.7, Java language, Java, Java Media Framework API 2.0, Mini-SQL1.0.16 and multimedia equipments. As results for verifying this platform functionally, we showed executing results of each module we mentioned above, and a numerical data obtained from QoS control procedures on client and server's GUI, while stream service is executing on our platform.

※ 본 연구는 2000년 정통부 대학기초 연구사업(2000-067-01)의 지원으로 수행되었음.

† 준 회 원 : 원광대학교 대학원 컴퓨터공학과

†† 정 회 원 : 원광대학교 컴퓨터공학과 교수

논문접수 : 2000년 10월 30일, 심사완료 : 2000년 12월 21일

1. 서 론

최근 인터넷이 보편화됨에 따라 원격교육, 원격진료, 공동작업, 화상회의, AOD, VOD 등의 분산 멀티미디어 서비스 분야에 관심이 증대되고 있다. 그러나 LAN, 인트라넷, 익스트라넷 및 인터넷과 같은 패킷-기반의 망 환경에서 사용자 수의 증가와 새로운 멀티미디어 데이터의 출현은 이용 트래픽의 급증을 유발하였고 이로 인한 지연(delay), 지연변이(delay variation, jitter), 패킷 손실(packet loss), 네트워크의 혼잡(congestion) 등과 같은 문제점을 낳고 있다. 이러한 문제점을 해결하고자 정보통신분야의 여러 표준화 기구는 인터넷 환경에서 QoS 보장이 가능하도록 RSVP, DiffServ, MPLS과 같은 새로운 프로토콜을 제안하고 있다[1, 2, 3]. 또한, 랜케스터(Lancaster), 콜롬비아(Columbia) 대학과 Tenet 등의 연구그룹을 통해 다양한 QoS 아키텍처가 제시되었다[4]. 그리고 객체 지향 개념의 등장으로 발전하게된 분산 객체 기술(CORBA, DCOM, Java 등)을 이용하여 분산 멀티미디어 서비스를 지원할 수 있는 모델들이 제안되었다. 그러나 멀티미디어 스트림 서비스를 제공하기 위한 프로토콜이나 QoS 아키텍처들은 특정 네트웍 환경에 적용하는 기술들이 대부분이며, IP기반의 네트워크 망에서 서비스 질을 보장하기 위해서는 별도의 추가적인 솔루션이나 장비를 필요로 한다. 그러므로 기능이 한정적이며, 확장성이나 재사용성의 문제점을 가지고 있다.

따라서, 본 논문에서는 분산 객체 기술인 CORBA를 이용하여 플랫폼에 독립적이며, 확장이나 재사용이 용이한 객체 서비스 형태의 QoS 통합 플랫폼을 제안하고자 한다. 이를 위한 QoS 제어 기법은 어플리케이션 레벨의 협약 기법[5]과 자원 모니터링을 통해 동적 적응이 이루어지도록 하였다. 또한, 스트림 송수신 객체와 QoS 통합 플랫폼을 서로 독립적으로 설계하고, API나 스크립트 형태의 제어방법을 택하였다.

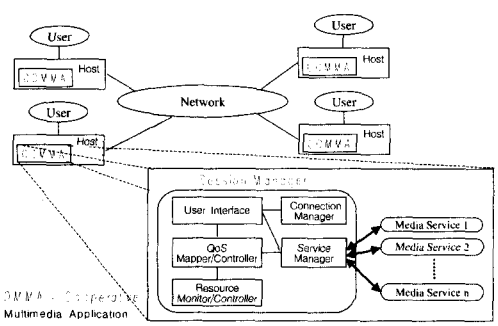
본 논문의 2장에서 관련된 연구들에 대해 알아보고, 3장에서 본 논문에서 제안하고 있는 플랫폼을 위한 고려 사항과 QoS 제어 기법에 대해 기술하며, 통합 플랫폼의 구조를 정의한다. 다음 4장에서는 제시한 QoS 통합 플랫폼을 구성하는 각 객체의 구조, 기능 그리고 서비스 수행절차를 UML (Unified Modeling Language)을 이용하여 설계하였다. 5장에서 구현 환경 및 시연환경에 대해 기술하고 끝으로 6장에서 본 논문에서 제시한 플랫폼의 문제점 및 앞으로의 연구 방향에 대해서 기술한다.

2. 관련 연구

본 장에서는 제시하고자 하는 QoS 통합 플랫폼의 기초가 되는 대표적인 연구로 ICSI(International Computer Science Institute)의 COMMA을 이용한 CME 구조[6]와 BBN의 프로젝트로 개발되고 있는 QuO[7] 그리고 OMG에서 CORBA 오디오/비디오 스트리밍 표준[8, 9]에 대해 기술한다.

2.1 CME(Collaborative Multimedia Environment)

CME는 ICSI에서 개발한 COMMA(COoperative MultiMedia Application)을 이용하여 Mbone망에서 원격의 사용자들간에 공동 작업이 가능하도록 개발된 프로토타입 서비스 환경이다. CME의 특징은 사용자가 직접 서비스 질(QoS)에 대한 요구 사항을 미디어 서비스에 설정할 수 있도록 구현된 사용자 위주의 서비스 질(QoS) 제어 방법을 제공하며, 공동 작업을 원하는 다른 사용자를 초대하고, 미디어 서비스를 시작할 수 있는 인터페이스인 접속 인터페이스(Connection Interface), 사용자가 서비스 질(QoS)을 미디어 서비스에 설정할 수 있도록 하는 제어 인터페이스(Control Interface) 그리고 사용자의 요구에 따라 COMMA에 의해 자원을 관리하는 기능을 제공한다. 미디어 서비스로 사용되는 어플리케이션은 프리웨어 형태로 Mbone망에서 사용할 수 있는 vic, vat, wb를 이용하여 구축되었다. 전체적인 구조는 (그림 1)과 같다.

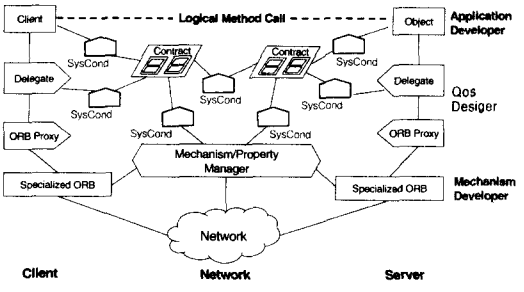


(그림 1) CME의 구조

2.2 QuO(Quality Object)

QuO는 컴퓨터 환경이나 네트워크 망의 변화에 영향

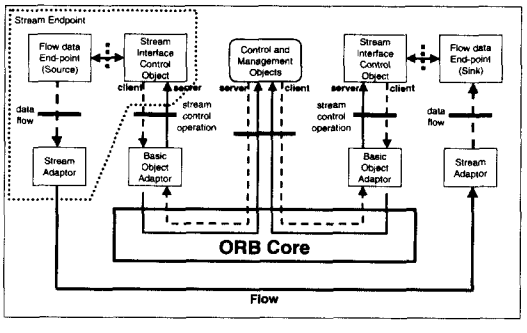
을 받지 않는 유연한 분산 어플리케이션을 작성하고, 재사용이 가능한 향상된 미들웨어 환경을 제공하는데 목적을 두고 있다. 이를 상세하게 기술하면, QuO는 분산 객체 미들웨어 기반에서 접속 설정, 서비스 요구의 변화 및 자원의 가용성 변화에 동적으로 실시간 적응이 가능한 확장된 소프트웨어 개발 프레임워크이다. QuO는 인터넷을 포함하는 WAN망에서 분산된 객체로 구성된 소프트웨어 어플리케이션간에 서비스 질(QoS)을 제어하는 환경을 제공한다. QuO 구조는 다음 (그림 2)와 같다.



(그림 2) QuO의 개발 환경

2.3 OMG의 오디오/비디오 스트리밍 서비스

OMG에서는 기존의 CORBA 구조를 변경하지 않고 오디오/비디오 스트림을 제어하기 위한 표준 인터페이스를 정의하였으며, 멀티미디어 데이터의 전송은 CORBA GIOP/IIOP를 사용하지 않고 멀티미디어 데이터 전송에 적합한 전송 프로토콜을 사용하도록 설계되어 있다. OMG 스트림 서비스 구조는 (그림 3)과 같다.



(그림 3) OMG 스트림 구조

관련된 대부분 연구에서 제안하고 있는 프레임워크들은 멀티미디어 응용을 위해 각 개발회사 자체의 스트림 설정/제어 메커니즘을 사용하기 때문에 서로 다른 하드웨어 플랫폼, 운영체제, 네트워크 환경에서 상호운용성이 어렵다. 더욱이 현재 개발된 스트림 프레임워크 자체가 분산처리를 지원할 수 있는 구조는 아니다. 즉 한 컴퓨터 시스템 내에 존재하는 소프트웨어 구성요소(미디어장치, CODEC, 통신채널 등)들은 통합적인 제어 및 관리가 가능하지만, 이들 구성요소들 중 일부가 네트워크상의 임의의 컴퓨터 시스템 내에 존재할 경우, 이를 통합적으로 제어하고, 관리하기는 어렵다. 따라서 이러한 분산된 이 기종의 시스템간에 통합된 자원 관리 방법이 필요하다. 다음 <표 1>은 이러한 관련연구에 대한 특징을 나타내고 있다.

<표 1> 관련 연구 특징

분 류	특 징	
CME	Mbone망에서 공동작업	COMMA 이용
QuO	인터넷을 포함하는 WAN 망	기존 ORB의 확장
OMG	멀티미디어 전송을 위한 전송 프로토콜 이용	표준 인터페이스 정의

3. QoS 통합 플랫폼

본 장에서는 관련 연구의 분석을 통해 스트림 서비스 구축시 고려할 사항과 QoS 파라미터에 따른 분류에 대해 기술하고, 이를 기반으로 QoS를 제어하는 다양한 기법에 대해 알아본다. 그리고 분산 환경에서 멀티미디어 데이터를 처리하는 스트림 서비스에 이용 가능한 QoS 통합 플랫폼을 제시한다.

3.1 고려 사항

스트림 서비스란 인터넷 또는 인트라넷에서 오디오/비디오 등의 멀티미디어 데이터를 다운로드가 아닌 실시간으로 주고받는 것을 말한다. 이러한 멀티미디어 데이터의 전송은 네트워크에 가해지는 트래픽이 지속적이고, 많은 네트워크 및 시스템 자원을 요구한다. 또한, 대부분의 멀티미디어 데이터는 스트림 서비스의 수요자(Consumer)와 제공자(Provider)간에 상호작용이 중요하며, 이를 위해 네트워크를 통한 연속적인 스트림 형태로 이루어진다. 이러한 스트림 서비스 구축 시에 고려할 사항들에 대해 알아보면 다음과 같다.

- QoS 제어 기능을 제공.
- 멀티 스트림 전송하는 경우 동기화 기법을 제공.

- 다양한 표현 방식 제공 : 다양한 재생 방법을 요구한다. 시작(start), 정지(stop), 일시정지(pause), 확대/축소(zoom in/out)와 같은 사용자 제어 방법을 제공.
- 네트워크 망에서 분산된 다양한 시스템간의 서비스 제공.
- 스트림 데이터의 양을 줄여 대역폭을 절약할 수 있도록 데이터의 압축을 제공.

기존의 서비스 질의 개념은 대부분 네트워크 서비스 레벨에서의 측정을 기초로 하였다. 즉, 네트워크의 지연 시간이나 데이터 손실률 등의 측정을 통해 실현하게 된다. 그러나 최근에 서비스 질에 대한 개념은 종단 시스템(End-to-End Systems)간에 서비스의 개념으로 확대되었다. 서비스 수행 과정에서 수요자와 서비스 제공자간에 서비스수준협약(SLA)을 통해 약속된 등급의 서비스를 유지하도록 제어하는 모든 작업을 의미한다. 이러한 서비스 질은 자원 관리라는 입장에서 보면, 사용자에게 얼마나 많은 자원을 할당하여 서비스를 제공하는가의 자원 배분에 대한 문제로 해석할 수 있다. 서비스 품질을 제어하기 위해서 자원을 분류하게 되는데, 이 분류된 자원을 QoS 파라미터(Parameter)라 한다. QoS 파라미터의 특성에 따라 상세히 분류해 보면 크게 세 가지로 나누어진다[10, 11].

- **사용자 레벨 QoS(User-Level QoS)**
  사용자의 주관적인 면이 많이 반영되는 관점으로 서비스 상태의 좋고, 나쁨을 나타내는 기준을 말한다. 일반적으로 5단계(excellent, good, fair, poor, bad)로 표현한다.
- **어플리케이션 레벨 QoS(Application-Level QoS)**
  스트림을 송수신하는 어플리케이션 관점에서 고려되는 요소(factor)들로 해상도, 프레임 전송률 등으로 표현한다.
- **자원 레벨 QoS(Resource-Level QoS)**
  가장 하위 레벨로 주로 하드웨어에 관련된 시스템 및 네트워크 자원 정보들로 전송률, 지연, 대역폭 등으로 표현한다.

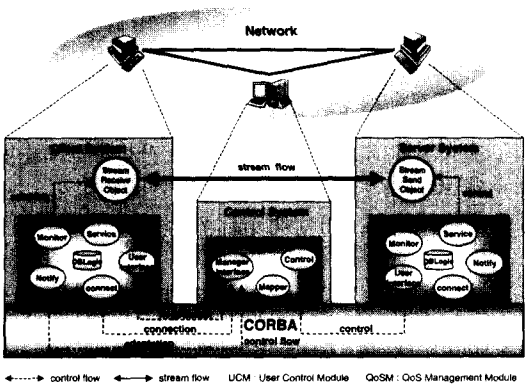
위에서 설명한 각 레벨간에 매핑을 위한 관리 정보를 QoS MIB (Management Information Base)라고 한다. 인터넷상에 연결된 여러 시스템들이 네트워크 대역폭을 공유하기 때문에 가용량(capacity)이 가변적이고, 네트워 트래픽이 늘어날수록 가용한 대역폭이 제한된다. 또한 인터넷망의 트래픽을 예측하여, 서비스를 동적으로 네트워크 환경에 대처할 수 있는 적응 기법이 요구된다[12]. 즉, 멀티미디어 서비스를 이용하는 사용자들은 서비스 제공자들에게 다양한 요구를 하게 되며, 이러한 서로 다른 사용자의 요구 조건을 만족시키기 위해서 QoS 제어 기능이 필수적이다[13]. 이러한 QoS 제어 기법은 다음과 같다.

- **표현/매핑(Specification and Mapping)**
  사용자의 요구 조건을 기술하는 표기법과 사용자의 서비스 요구 조건을 어플리케이션이나 시스템이 이해할 수 있도록 변환하는 기능을 제공한다.
- **협약(Negotiation)**
  사용자와 서비스 시스템간의 적합한 QoS 파라미터 값을 찾아내는 기능을 제공해야 한다. 즉 사용자가 요구하는 서비스의 제공 가능 여부를 판단하여 시스템이 사용자에게 알리는 기능을 지원한다.
- **자원 적응(Resource Adaptation)**
  가변적인 네트워크 환경이나 시스템 자원의 가용량에 따라 동적으로 서비스 품질을 제어할 수 있는 기법을 제공한다[14, 15].
- **모니터링/통보(Monitoring/Notification)**
  현재 서비스 상태를 감시하고 기록하는 기능으로 이를 통해 사용자는 또 다른 품질의 서비스를 요구할 수 있으며, 서비스 위반 상황을 알릴 수 있는 기능을 제공한다.

앞서 설명한 관리 기능 외에 서비스협약수준에 따라 멀티캐스팅이나 다자간 서비스 환경을 제공하기 위해 사용되는 승인 제어, 자원 예약, 스케쥴링, 정책 등이 요구된다. 본 논문에서 제안하고 있는 QoS 통합 플랫폼은 어플리케이션 레벨에서의 서비스 질의 표현, 매핑, 협약, 자원 적응, 자원 모니터링, 통보 기법을 통해 스트림 서비스를 관리한다.

3.2 QoS 통합 플랫폼 구조

제안하는 QoS 통합 플랫폼은 (그림 4)와 같이 중앙 제어방식으로 종단간에 스트림 흐름을 관리하며, 관련 있는 객체를 묶기 위한 객체 모듈(Object Module)과 스트림 송수신 객체 또는 프로그램으로 구성된다.



(그림 4) QoS 통합 플랫폼

객체 모듈은 관련된 객체를 패키지 단위로 정의하며, 하나이상의 객체와 이들 객체를 관리하는 인터페이스 객체로 구성된다. 또한 스트림 송수신 시스템간에 서비스 질을 보장할 수 있도록 상호 작용하는 두 개의 사용자 제어 모듈과 하나의 QoS 관리 모듈을 객체 모듈로 정의한다. 이 객체 모듈을 구성하는 모든 객체는 CORBA 객체이다. QoS 통합 플랫폼을 구성하는 다른 구성요소로 스트림 송수신 객체(Stream Send/Receive Object)는 스트림 데이터의 전송과 수신을 담당하며, QoS 관리 플랫폼과 별도로 개발되어 서비스 객체가 관리하며, 기존에 개발된 화상회의 툴(vic, vat)이나 VOD 시스템을 수정 없이 스트림 인터페이스를 사용하도록 설계한다. 제안된 QoS 통합 플랫폼에서 제어 흐름(control flow)은 ORB을 이용하고, 스트림 흐름(stream flow)은 IP기반에서 TCP나 UDP상에 RTP 패킷을 송·수신하도록 스트림 송수신 객체에서 처리한다.

4. QoS 통합 플랫폼 설계

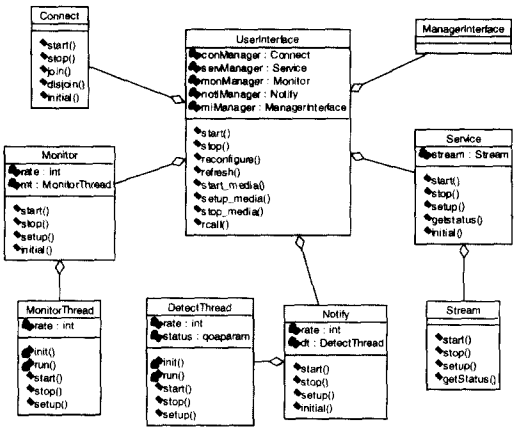
본 장에서는 3장에서 제시한 QoS 통합 플랫폼을 이루는 객체의 세부 속성 및 메소드 설계를 클래스 다이어그램(Class Diagram)을 사용하여 객체 구조를 정의하고, 데이터 흐름도(Data Flow Diagram)를 이용하여 객체간에 데이터의 흐름을 나타내며, 시퀀스 다이어그램(Sequence Diagram)을 이용하여 접속, 협약, 자원 적응, 자원 모니터링, 접속 해제 절차를 기술한다. 이에 대한 모델링은 UML를 사용하였다.

4.1 객체 구조 설계

제안한 QoS 통합 플랫폼을 구성하는 객체는 사용자 제어 모듈과 QoS 관리 모듈로 나누어 객체들의 구조를 UML의 클래스 다이어그램으로 설계하였다.

4.1.1 사용자 제어 모듈(User Control Module)

사용자 제어 모듈은 사용자 인터페이스 객체, 접속 객체, 서비스 객체, 통보 객체, 모니터 객체와 DB 로직으로 구성된다. 이에 대한 클래스 다이어그램은 (그림 5)와 같다.



(그림 5) 사용자 제어 모듈

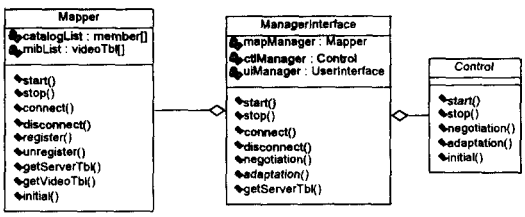
- **사용자 인터페이스 객체(UserInterface)**는 스트림 응용 프로그램의 개발에 사용할 수 있는 인터페이스 제공과 사용자 제어 모듈을 구성하는 객체에 대한 관리를 담당한다.
- **접속 객체(Connect)**는 스트림 서비스를 원하는 시스템간에 연결 설정 및 해제 기능을 수행하며, 매퍼 객체로부터 접속 가능한 서버 목록인 시스템 카탈로그를 통해 서비스를 설정한다.
- **서비스 객체(Service)**는 스트림 전송을 담당하는 스트림 객체를 관리하는 객체로 속성 정보로 스트림 객체에 대한 객체 참조자를 가지며, 메소드는 스트림 객체의 생성, 서비스 시작, 종료, 재설정을 위한 메소드를 제공한다.
- **모니터 객체(Monitor)**는 스트림 객체로부터 스트림 데이터의 수신 상태와 서버 시스템의 자원 사용량을 DB에 기록하는 모니터 스레드와 모니터링 주기를 나타내는 속성으로 구성된다. 그리고 모니터링

에 관련된 메소드는 스레드의 제어 기능을 수행한다.

- **통보 객체(Notify)**는 모니터 객체에서 기록해 놓은 모니터링 데이터를 분석하여 스트림 서비스 종단간에 협약된 QoS가 유지되는지를 감시하며, 모니터 객체와 마찬가지로 스레드를 통해 수행하도록 스레드 객체의 참조자를 유지한다.

4.1.2 QoS 관리 모듈(QoS Management Module)

다음 (그림 6)에서 나타난 바와 같이 QoS 관리 모듈은 관리자 인터페이스 객체, 매퍼 객체, 제어 객체로 구성된다.



(그림 6) QoS 관리 모듈

- **관리자 인터페이스 객체(ManagerInterface)**는 구성 객체들을 통합적으로 관리하며, QoS 관리 플랫폼에서 사용하는 여러 자료구조를 관리하는 메소드를 제공한다.
- **제어 객체(Control)**는 사용자가 요청하는 협약, 적응 메시지를 처리하는 객체로 매퍼 객체의 QoS MIB을 기초로 종단간에 서비스 질을 유지하는 기능을 제공한다.
- **매퍼 객체(Mapper)**는 접속 가능한 시스템 카탈로그를 제공하며, 스트림 서비스가 이루어지는 두 사용자 제어 모듈간을 접속을 설정하고, 해제하는 기능을 제공한다. 그리고 제어 객체에서 서비스 질을 유지하는데 사용되는 관리 정보(MIB) 목록을 관리하는 기능을 수행한다.

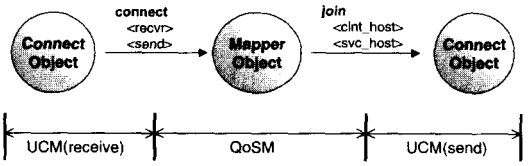
4.2 객체간에 관계 및 기능 설계

본 절에서는 객체간에 관계를 중심으로 제안하는 QoS 통합 플랫폼에서 제공되는 QoS 제어 기법을 객체간에 데이터의 흐름을 통해 설명하며, 이를 통해 각 객체의 구체적인 기능 및 역할을 정의한다.

4.2.1 접속 절차

(그림 7)은 종단간에 스트림 서비스 시작을 위해 사용자 제어 모듈간에 접속 과정과 함께 객체간에 메시지와 데이터 흐름을 나타낸다. 수행 절차는 다음과 같다.

- 수신측 접속 객체가 QoS 관리 모듈내의 매퍼 객체에게 연결 요청 메시지와 접속할 사용자 정보를 함께 보낸다.
- 매퍼 객체는 서비스 연결을 위해 송신측 접속 객체에게 join 메시지를 보낸다.
- 송신측 접속 객체가 스트림 서비스 접속을 승인함으로써 접속 절차가 이루어진다.

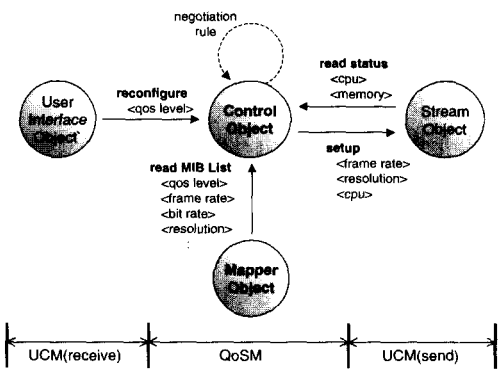


(그림 7) 접속 절차의 데이터 흐름도

4.2.2 협약 기법

협약 기법은 (그림 8)과 같이, 데이터의 교환이 이루어지며, 관련된 객체로는 사용자의 재설정 메시지를 전달하는 사용자 인터페이스 객체와 협약 알고리즘을 수행하는 제어 객체, 협약에 사용되는 관리 정보를 제공하는 매퍼 객체 그리고 협약된 서비스 레벨을 제공하는 스트림 송신 객체로 구성된다. 협약 기법의 수행 절차는 다음과 같다.

- GUI을 이용해 사용자가 원하는 QoS의 등급을 제어 객체에게 전달함으로써 서비스 재설정을 요구한다.
- 사용자의 요구를 받은 제어 객체는 매퍼 객체에게



(그림 8) 협약 기법 절차의 데이터 흐름도

서비스 관리 정보인 QoS MIB을 요청한다.

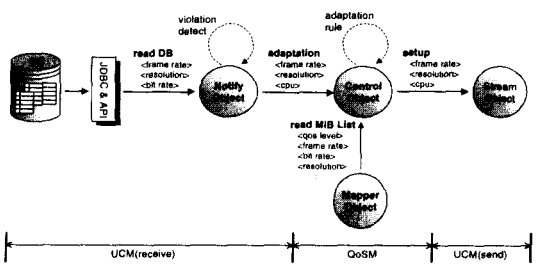
- 매퍼 객체는 요청한 QoS MIB을 제어 객체에게 넘긴다.
- 제어객체에 기술된 협약 알고리즘에 따라 서비스 제공 여부를 판단한다.
- 만약 협약이 성공한 경우, 제어 객체가 스트림 전송 객체에게 협약된 서비스가 가능하도록 QoS 파라미터를 포함하여 재설정 메시지를 보낸다.
- 재설정 메시지를 받은 스트림 전송 객체는 전달된 파라미터로 재설정한다.

4.2.3 자원 적응 기법

자원 적응 기법은 스트림 서비스 과정에서 발생하는 예외 상황이나 서비스 질의 유지가 어려운 경우에 적절한 서비스 질로 낮추어서 종단간에 서비스의 중단을 방지하기 위한 목적으로 이용된다. 적응 기법은 (그림 9)의 객체간의 메시지와 데이터의 교환으로 수행된다.

객체들간에 자원 적응 절차는 다음과 같다.

- 모니터 객체에 의해 기록된 서비스 상태 정보를 DB로부터 읽는다.
- DB에서 읽은 데이터를 분석하여 협약된 QoS 위반 여부를 검사한다.
- 만약 위반 사항이 검출되면, QoS 관리 모듈의 제어 객체에게 Adaptation 메시지를 통해 자원 적응 기법을 요청한다.
- QoS 관리 모듈내의 제어 객체에서 현재 서비스되는 파라미터 정보를 기초로 설정되어 있는 QoS MIB에서 적절한 서비스 레벨을 선정한다.
- 선정된 레벨에 맞는 파라미터 값으로 스트림 전송 객체를 재설정한다.

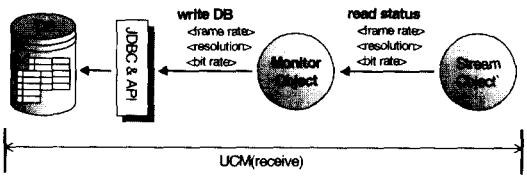


(그림 9) 자원 적응 기법 절차의 데이터 흐름도

4.2.4. 자원 모니터링 기법

자원 모니터링은 스트림 서비스의 유지에 이용되는 QoS 파라미터를 모아 DB에 저장하는 과정을 말하며, 모니터링 절차는 (그림 10)과 같이 메시지와 데이터의 교환으로 수행된다. 자원 모니터링 기법의 수행 절차는 다음과 같다.

- 모니터 객체가 주기적으로 스트림 수신 상태(프레임 전송률, 해상도) 및 시스템 자원(CPU 사용량)의 소비량을 검출한다.
- 검출된 내용을 DBMS 모듈이나 전용 API을 통해 DB 테이블에 기록한다.
- 위 과정을 반복적으로 수행한다.

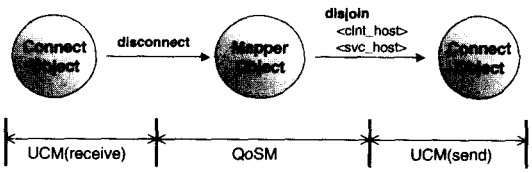


(그림 10) 자원 모니터링 절차의 데이터 흐름도

4.2.5 접속 해제 절차

(그림 11)은 스트림 서비스 접속 해제 절차를 사용자 제어 모듈간에 접속 해제 절차에서 메시지와 데이터의 교환을 나타낸다. 접속 해제 절차는 다음과 같다.

- 수신측 접속 객체가 QoS 관리 모듈내의 매퍼 객체에게 연결 해제 메시지를 접속 해제할 사용자 정보와 함께 요청한다.
- 매퍼 객체는 접속 해제할 송신측 접속 객체에게 disjoin 메시지를 보낸다.
- 송신측 접속 객체는 스트림 송신을 정지시키고, 수신측 접속 객체에게 이를 통보함으로써 접속 해제가 이루어진다.



(그림 11) 접속 해제 절차의 데이터 흐름도

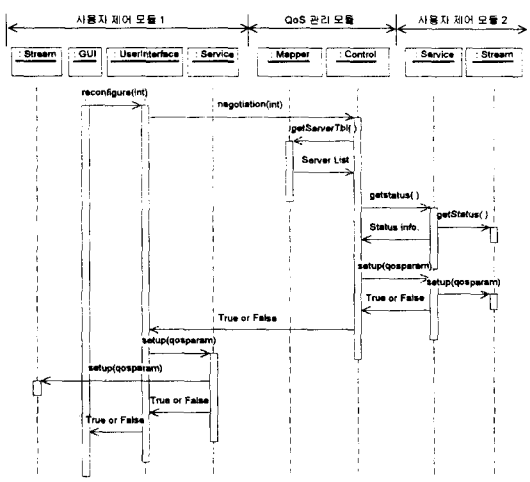
4.3 시나리오 설계

본 논문에서 제시한 플랫폼의 수행 과정 시나리오는 서비스 접속, 해제, 협약, 자원 적응 및 자원 모니터링

절차로 나누어, 시퀀스 다이어그램을 통해 객체간에 시간 변화에 따른 메시지의 흐름으로 표현한다. 시퀀스 다이어그램에서 사용자 제어 모듈1은 스트림 수신 객체를 제어하고 사용자 제어 모듈2가 스트림 전송 객체를 제어하는 상황에서 종단 시스템간에 서비스를 유지하는 과정을 사용자 제어 모듈과 QoS 관리 모듈을 구성하는 객체들의 상호작용을 통해 나타낸다.

4.3.1. 접속 절차

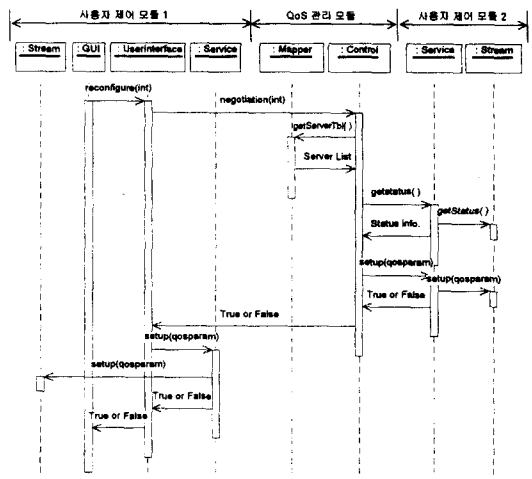
서비스 종단간의 스트림 서비스를 시작하는 과정은 사용자가 사용자 제어 모듈1의 사용자 인터페이스 객체의 start() 오퍼레이션을 통해 접속 객체에게 사용자 제어 모듈2의 접속 객체에게 접속을 요청한다. 만약 두 사용자 제어 모듈간에 접속이 성공하면, 사용자 제어 모듈2의 스트림 전송 객체는 데이터 전송을 시작하고, 사용자 제어 모듈1은 스트림 수신 객체를 통해 스트림 데이터를 수신하도록 서비스 객체의 오퍼레이션을 통해 스트림 수신 객체를 구동시킨다. 그리고 스트림 수신 객체의 오퍼레이션을 통해 스트림 수신 상태를 모니터링하고 감시하도록 모니터 객체와 통보 객체를 구동시킴으로써 서비스가 접속된다. 종단간에 서비스 접속 절차는 (그림 12)와 같다.



(그림 12) 서비스 접속 절차의 시퀀스 다이어그램

4.3.2 협약 절차

협약 절차는 사용자 인터페이스 객체를 통해 QoS 관리 모듈의 제어 객체를 통해 종단간에 협약을 요청한다. 다음 (그림 13)은 이에 대한 수행 절차를 나타낸다.



(그림 13) 협약 절차의 시퀀스 다이어그램

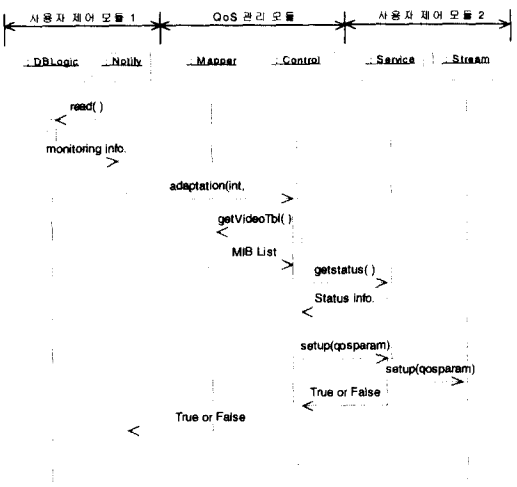
사용자가 요구하는 QoS 정보와 함께 협약 메시지를 받은 QoS 관리 모듈의 관리자 인터페이스 객체는 제어 객체에게 사용자가 요구한 QoS에 관한 정보를 넘기고 제어 객체는 매퍼 객체에게 QoS MIB를 요청하고 제공받은 QoS MIB와 사용자가 요구한 협약 정보를 가지고 협약 알고리즘을 수행하게 된다. 협약이 이루어지면, 사용자 제어 모듈2의 서비스 객체에게 협약된 정보에 맞게 스트림 송신 객체를 재설정하도록 요청한다. 스트림 송신 객체의 재설정이 끝나면 스트림 전송 형식에 맞도록 스트림 수신 객체의 재설정을 통해 서비스 종단간에 재설정된 QoS로 서비스가 수행된다.

4.3.3 자원 적응 절차

(그림 14)는 협약된 QoS 위반 상황이 발생했을 때, 서비스 종단간에 자원 적응 절차를 나타낸다.

네트워크 자원의 변화에 대처하기 위해 사용되는 자원 적응 절차는 사용자 제어 모듈1의 통보 객체가 협약된 서비스 질이 보장되지 못하는 조건을 검출함으로써 시작된다. QoS 관리 모듈에게 현재의 서비스 상태 정보와 함께 자원 적응 알고리즘을 수행하도록 QoS 관리 모듈의 제어 객체에게 adaptation 메시지를 보낸다. 이 메시지를 받은 제어 객체는 스트림 수신 객체에서 얻은 서비스 수신 상태 정보를 참고하여 사용자 제어 모듈2의 서비스 객체에서 관리하는 스트림 송신 객체를 재설정하도록 사용자 제어 모듈2의 서비스 객체에게 요청을 한다. 서비스 객체에 의해 스트림 송신 객체의 재설정을 통해 서비스 종단간에 자원 적응이

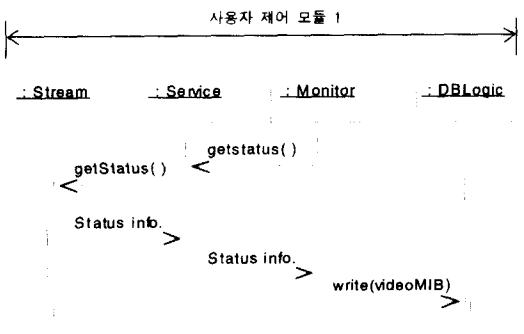
수행된다.



(그림 14) 자원 적응 절차의 시퀀스 다이어그램

4.3.4 자원 모니터링 절차

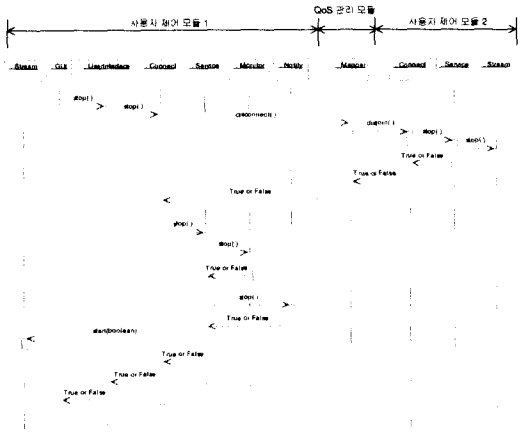
자원 모니터링 절차는 서비스 객체에 의해 실행된 모니터 객체는 서비스 객체로부터 스트림 수신 상태 정보를 요청하고 서비스 객체는 다시 스트림 객체로부터 수신 상태 정보를 얻는다. 이렇게 획득된 정보를 DB 테이블에 기록한다. 이 과정을 반복하여 모니터링이 수행된다. 다음 (그림 15)는 스트림 수신 상태를 모니터링하는 절차를 나타낸다.



(그림 15) 자원 모니터링 절차의 시퀀스 다이어그램

4.3.5 접속 해제 절차

접속 해제 절차는 사용자가 GUI 화면에 정지 버튼을 통해 사용자 인터페이스 객체에게 서비스 종료시키기 위해 접속 객체를 통해 QoS 관리 모듈의 매퍼 객체에게 접속 해제를 요청하고, 매퍼 객체는 다시 송신측 접속 객체에게 요청하여 서비스 객체를 통해 스트림 객체를 종료시킨다. 이후 사용자 제어 모듈의 서비스 객체를 통해 모니터 객체와 통보 객체를 정지시키고, 마지막으로 스트림 객체를 정지시켜 종단간에 스트림 서비스의 접속을 해제한다. (그림 16)은 서비스 접속 해제 절차를 나타내는 시퀀스 다이어그램이다.



(그림 16) 접속 해제 절차의 시퀀스 다이어그램

5. 구현 및 시연 환경

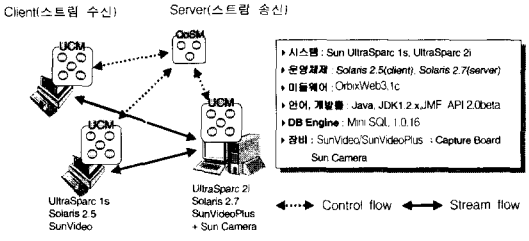
본 장에서는 구현에 관련된 제반 환경에 대해 설명하고, 설계된 객체를 CORBA 객체화하기 위한 IDL 설계 그리고 DB 스키마 설계, QoS MIB 설계, 스트림 객체에 구현에 관련된 내용을 중심으로 상세히 설명하고, 마지막으로 구현된 실행 화면을 중심으로 서비스 수행 절차를 보인다.

5.1 개발 환경

다음 (그림 17)은 개발환경으로 서버측 시스템에 QoS 관리 모듈과 사용자 제어 모듈을 등록하였고, 클라이언트측 시스템에 사용자 제어 모듈을 등록하여 클라이언트와 서버간에 1 대 1 스트림 송수신이 가능하도록 구현하였다.

스트림 수신 및 사용자 제어 인터페이스를 담당하는 시스템은 Sun Microsystems사의 UltraSparc 1s이며, 스트림을 송신하는 서버 시스템은 Sun UltraSparc 2i

와 실시간으로 카메라가 설치되어 입력된 영상을 Capture하여 클라이언트측에 송신하는 기능을 수행한다.



(그림 17) 개발 환경

QoS 통합 플랫폼에서 제어 메시지의 흐름을 담당하는 미들웨어로 IONA사의 OrbixWeb 3.1c를 사용하였으며, 클라이언트와 서버 시스템에 설치되어 종단간에 제어 메시지를 통해 스트림 서비스의 서비스 질을 유지한다. 스트림 데이터인 Capture한 이미지를 JPEG, H261, H263등의 형식으로 인코딩하여 TCP/UDP위에 RTP 패킷으로 전송하고, 디코딩하여 디스플레이 하기 위해 SunSoft사의 JMF API 2.0 beta2[16]을 이용해 구현하였다. 또한, 그래픽 사용자 인터페이스는 자바의 경량 컴포넌트인 JFC/Swing을 이용하여 구현하였다. 서버 시스템에 사용된 Capture 장비는 스트림 송신을 담당하는 서버 시스템에 SunVideoPlus Capture 보드와 Sun용 카메라를 사용하였으며, 클라이언트 시스템에 Sun Video Capture 보드를 사용하였다.

5.2 IDL 정의

본 절에서는 제시한 통합 플랫폼을 구성하는 객체들을 구현하기 위해 OMG IDL을 정의하였으며, 이 IDL을 중심으로 구현될 각 객체를 구성하는 애트리뷰트와 오퍼레이션에 대해 설명한다. QoS 관리 모듈 객체간에 공통적으로 사용되는 자료구조, 사용자 제어 모듈을 구성하는 접속 객체, 서비스 객체 그리고 사용자 인터페이스 객체에 대한 IDL을 정의한다. 다음으로 QoS 관리 모듈의 제어 객체, 매퍼 객체, 관리자 인터페이스 객체의 IDL을 정의한다.

5.2.1 자료 구조 정의

먼저 IDL에서 파라미터로 사용되는 자료 구조에 대한 정의를 보면 다음 (그림 18)과 같다.

```
struct member { // 사용자 정보 구조
  string name;
  string ip;
  string hostname;
};
struct videoMIB { // Video QoS MIB 구조
  long    level;
  float   framerate;
  string  encode;
  string  resolution;
  float   cpu;
  long    buffersize;
  long    packetsize;
};
struct qosparam { // 서비스 상태를 나타내는 주요 요소
  float   framerate;
  long    resolution;
  float   cpu;
};
typedef sequence<member> catalogList; //catalog, QoS MIB 목록
typedef sequence<videoMIB> mibList;
// forwarding declaration
interface ManagerInterface;
interface UserInterface;
```

(그림 18) 자료 구조

구조체 member는 종단간에 스트림 서비스 연결 과정에서 전달되는 접속 시스템에 대한 사용자 정보로 사용자 이름, 접속 시스템의 IP주소와 호스트 이름으로 구성된다. 접속 객체와 매퍼 객체간에 메시지 파라미터로 사용된다. videoMIB 구조체는 스트림 서비스 제어하는데 협약 및 자원 적응 기법에 사용될 정보로 구성은 서비스 질, 프레임 전송률, 인코딩 형식, 해상도, CPU 사용량, 버퍼 크기, 패킷 크기로 이루어져 있다. 다음으로 qosparam 구조체는 종단간에 협약이나 자원 적응 알고리즘 수행 과정에서 스트림 서비스의 상태 정보를 나타내기 위해 메시지와 같이 전달되는 파라미터로 사용되며, 프레임 전송률, 해상도, CPU 사용량으로 구성된다. 마지막으로 접속 가능한 서버 목록을 나타내는 카탈로그인 catalogList와 QoS MIB 목록인 mibList의 두 가지 목록 정보를 갖는다. 이 자료구조는 사용자 제어 모듈과 QoS 관리 모듈에서 공유하여 사용한다.

5.2.2 사용자 제어 모듈 IDL

● 사용자 인터페이스 객체

다음 (그림 19)는 사용자 인터페이스 객체의 IDL을 나타낸다.

```
interface UserInterface { // 사용자 인터페이스 객체의
IDL 정의
// Attributes
  attribute Connect conManager;
  attribute Service servManager;
  attribute ManagerInterface miManager;
// Operations
  boolean start(in string local, in string remote, in
string manager, in string dbport);
  boolean stop();
  boolean reconfigure(in long qoslevel);
  boolean refresh();
  boolean start_media(in qosparam param);
  boolean setup_media(in qosparam param);
  boolean stop_media();
  boolean rcall(in string local, in string remote, in
string manager);
};
```

(그림 19) 사용자 인터페이스 객체의 IDL

애트리뷰트를 살펴보면 사용자 제어 모듈을 구성하는 제어 객체, 서비스 객체, 관리자 인터페이스 객체에 대한 객체 참조자(Object Reference)를 관리하기 위해 conManager, servManager, miManager를 정의하였다. 서비스의 시작을 위해 start() 오퍼레이션, 종단간에 QoS 협약을 요청하기 위해 사용되는 reconfigure() 오퍼레이션, GUI 화면의 모니터링 시작을 위한 refresh() 오퍼레이션, 스트림 송신을 담당하는 사용자 제어 모듈의 스트림 송신 시작을 요청하는 start_media() 오퍼레이션, 스트림 송신 파라미터를 재설정하기 위한 setup_media() 오퍼레이션, 스트림 송신 정지를 위한 stop_media() 오퍼레이션을 제공하며, 마지막으로 송신측 사용자 제어 객체들의 초기화를 요청하는 rcall() 오퍼레이션을 제공한다.

● 접속 객체

접속 객체를 구성하는 오퍼레이션은 스트림 송신측 시스템에 접속하기 위해 사용되는 start() 오퍼레이션을 제공하며, 서비스 세션을 종료하기 위한 stop() 오퍼레이션, 세션 참여 여부를 알아보기 위해 매퍼 객체에 의해 호출되는 join() 오퍼레이션, 세션에서 탈퇴하는데 사용되는 disjoin() 오퍼레이션, 접속 객체의 전역 변수를 초기화를 위한 initial() 오퍼레이션을 제공한다. 아래의 (그림 20)은 접속 객체의 IDL을 나타낸다.

```
interface Connect { // 접속 객체의 IDL 정의
// Operations
  boolean start(in member receive, in member send);
  boolean stop();
  boolean join();
  boolean disjoin();
  boolean initial(in UserInterface ui, in string manager);
};
```

(그림 20) 접속 객체의 IDL

● 서비스 객체

본 논문의 3장에서 설계된 시퀀스 다이어그램을 보면, 서비스 객체와 모니터 객체, 통보 객체는 연관된 작업이 많다. 이로 인해 세 가지 객체들간에 많은 통신이 발생한다. 만일 이들 객체가 C++이나 자바를 사용해 만들어진 객체라면 이러한 통신은 지역 메소드 호출임으로 문제가 되지 않지만, 구현된 객체들이 CORBA 객체로 구성되어 있기 때문에 통신 오버헤드가 발생한다.

즉, 모니터 객체는 주기적으로 서비스 객체의 get-status() 오퍼레이션을 원격 메소드 호출하게 된다. 따라서 구현과정에서 성능저하를 해결하고자 통보 객체와 모니터 객체를 서비스 객체에 통합하여 통보 객체와 모니터 객체를 스레드로 구현하고 서비스 객체만을 CORBA객체로 구현한다. 서비스 객체에서 모니터 스레드와 통보 스레드를 재설정할 수 있도록 setup_notify() 오퍼레이션과 setup_ monitor() 오퍼레이션을 추가하여 서비스 객체의 IDL을 정의한다. 아래의 (그림 21)은 새로 정의된 서비스 객체를 나타낸다.

```
interface Service { // 서비스 객체의 IDL 정의
// Operations
  void start(in boolean isSend, in long srate, in long
qosparam param);
  boolean stop();
  boolean setup(in qosparam param);
  boolean setup_notify(in long rate, in qosparam param);
  boolean setup_monitor(in long rate);
  qosparam getstatus();
  boolean initial(in UserInterface ui, in string manager);
};
```

(그림 21) 서비스 객체의 IDL

서비스 객체의 오퍼레이션을 설명하면, 스트림 데이터를 수신하고 모니터링 하기 위해 스트림 객체를 초기화하고, 모니터 객체와 통보 객체를 시작시키는 start() 오퍼레이션을 제공하며, 스트림 객체를 정지시키고, 모니터 객체와 통보 객체를 정지시키기 위한

stop() 오퍼레이션을 제공하며, 스트림 객체의 서비스 상태(프레임 전송률, 해상도, CPU사용량)를 얻어오기 위한 getstatus() 오퍼레이션을 제공하며, 마지막으로 서비스 객체의 전역 변수를 초기화하기 위한 initial() 오퍼레이션을 제공한다.

5.2.3 QoS 관리 객체 모듈 IDL 설계

● 관리자 인터페이스 객체

아래 (그림 22)은 관리자 인터페이스 객체의 IDL을 나타낸다.

```
interface ManagerInterface { // 관리 인터페이스 객체의 IDL 정의
// Attributes
  attribute Mapper mapManager;
  attribute Control ctlManager;
  attribute UserInterface uiManager;
// Operations
  boolean start(in string local, in string remote, in string manager);
  boolean stop();
  boolean connect(in member receive, in member send);
  boolean disconnect();
  boolean negotiation(in long qoslevel);
  oolean adaptation(in long qoslevel, in qosparam param);
  catalogList getServerTbl();
};
```

(그림 22) 관리자 인터페이스 객체의 IDL

관리자 인터페이스 객체는 QoS 관리자 모듈을 구성하는 매퍼 객체와 제어 객체에 대한 객체 참조자를 관리하기 위해 mapManager, ctlManager, uiManager의 애트리뷰트를 유지하며, 매퍼 객체와 제어 객체를 초기화하기 위해 start() 오퍼레이션과 QoS 관리 모듈의 제어 서비스를 정지시키기 위해 stop() 오퍼레이션을 제공한다. 매퍼 객체에게 서비스 연결/해제를 요청하기 위해 connect() 오퍼레이션과 disconnect() 오퍼레이션을 제공하며, 제어 객체에게 협약과 자원 적응 알고리즘 수행을 요청하는 negotiation() 오퍼레이션과 adaptation() 오퍼레이션을 제공하며, 시스템 카탈로그 데이터를 제공하기 위해 getServerTbl() 오퍼레이션을 제공한다.

● 제어 객체

다음 (그림 23)은 제어 객체의 IDL을 나타낸다.

```
interface Control { // 제어 객체의 IDL 정의
// Operations
  boolean start();
  boolean stop();
  boolean negotiation(in long qoslevel);
  boolean adaptation(in long qoslevel, in qosparam param);
  boolean initial(in ManagerInterface mi, in string remote);
};
```

(그림 23) 제어 객체의 IDL

제어 객체는 객체를 활성화와 비활성화를 시키기 위한 start() 오퍼레이션과 stop() 오퍼레이션을 제공하며, 협약을 처리하는 negotiation() 오퍼레이션과 자원 적응을 처리하는 adaptation() 오퍼레이션을 제공한다. 제어 객체를 초기화하기 위해 initial() 오퍼레이션을 제공한다.

● 매퍼 객체

(그림 24)는 매퍼 객체의 IDL을 나타낸다

```
interface Mapper { // 매퍼 객체의 IDL 정의
// Attributes
  attribute catalogList serverTbl;
  attribute mibList videoTbl;
// Operations
  boolean start();
  boolean stop();
  boolean connect(in member receive, in member send);
  boolean disconnect();
  boolean register(in member mem);
  boolean unregister(in member mem);
  catalogList getServerTbl();
  mibList getVideoTbl();
  boolean initial(in ManagerInterface mi, in string remote);
};
```

(그림 24) 매퍼 객체의 IDL

매퍼 객체는 접속 가능한 시스템 목록을 관리하기 위해 애트리뷰트 catalogList와 협약과 자원 적응 기법에 사용되는 QoS MIB 목록을 관리하기 위한 애트리뷰트 mibList를 가진다. 매퍼 객체의 활성화와 비활성화를 위해 start()과 stop() 오퍼레이션을 제공하며, 사용자 제어 모듈의 접속 객체의 요청에 따라 세션에 참여시키고, 탈퇴시키기 위한 connect()과 disconnect() 오퍼레이션을 제공한다. 또한 카탈로그를 갱신하기 위한 register()과 unregister() 오퍼레이션을 제공한다. 그리고 제어 객체에게 QoS MIB 목록을 제공하기 위해 getVideoTbl() 오퍼레이션을 제공하며, 접속 객체에

게 카탈로그를 제공하기 위해 getServerTbl() 오퍼레이션을 제공한다. 마지막으로 매퍼 객체를 초기화하기 위해 initial() 오퍼레이션을 제공한다.

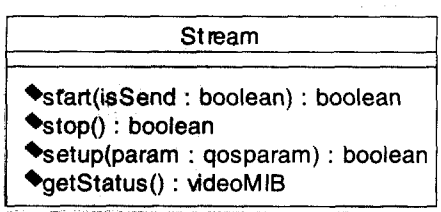
5.3 스트림 객체(Stream Object) 구현

스트림 객체는 비디오 스트림의 전송 기능만을 구현하였으며, 기본 이미지 인코딩 형식은 JPEG 형식을 사용하였다. TCP을 이용해 RTP 패킷으로 스트림 객체간에 전송하며, 320×240, 160×120의 두 가지 크기의 해상도를 지원하고, 스트림을 생성 전송하는 스트림 객체에서는 최대 30 프레임/초의 속도로 송수신이 가능하도록 구현한다. 스트림 객체의 인코딩 형식에 따른 해상도, CPU사용량, 필요로 하는 대역폭은 다음 <표 2>와 같다.

〈표 2〉 스트림 객체의 지원 가능한 미디어 형식

Format	Content Type	Quality	CPU Req.	Bandwidth Req.
H.261	AVI / RTP	Low	Medium	Medium
H.263	QuickTime AVI / RTP	Medium	Medium	Low
JPEG	**QuickTime AVI / RTP**	**High**	**High**	**High**

QoS 통합 플랫폼에서 스트림 서비스를 제어하기 위해 사용자 제어 모듈 객체 중 서비스 객체를 통해 스트림 객체를 제어하며, 현재 제공되는 인터페이스는 (그림 25)와 같다.



(그림 25) 스트림 객체의 인터페이스

스트림 객체 인터페이스 중에서 start() 오퍼레이션은 스트림 객체가 스트림 데이터를 보내거나 받을 수 있도록 객체를 초기화하고 구동시키며, stop() 오퍼레이션은 스트림 객체를 정지시키고 할당된 자원을 해제하는 기능을 수행한다. setup() 오퍼레이션은 스트림 객체의 서비스 상태를 조절할 수 있는 스트림 파라미터를 설정하는 인터페이스이다. 마지막으로 getStatus() 오퍼레이션은 모니터링을 위해 스트림 객체로부터 스트림 파라미터를 얻어오는 기능을 수행한다. 스트림 송수신 프로그램이 객체의 형태가 아닌 경우 서비스 객체에서 쉽게 제어하도록 스트림 송수신 프로그램을 포장하는 역할을 담당한다. 현재 구현된 스트림 객체는 public 형태의 메소드로 구현하였다. 그러나 기존에 개발된 비객체 형태의 프로그램을 제어하기 위해 스트림 객체 인터페이스를 통해 프로그램을 제어하도록 구현해야 한다.

5.4 DB 스키마 및 QoS MIB

본 절에서는 모니터링에 사용되는 관련 스키마 설계를 중심으로 구현에 사용된 DB 엔진, QoS MIB 설계에 대해 설명한다.

5.4.1 DB 엔진과 인터페이스

QoS 통합 플랫폼에서 모니터링 데이터를 저장하기 위해 DB 엔진으로 Mini SQL(mSQL) 1.0.16을 이용하였다. 또한, 인터페이스는 imaginary에서 제공하는 mSQL -JDBC 드라이버 2.0을 사용하여 구현하였다.

5.4.2 DB 스키마

클라이언트측의 스트림 객체로부터 서비스를 제공받는 동안 스트림 데이터 수신 상태 즉, 프레임 전송률, 인코딩 형식, 해상도, 전송률, 버퍼 크기, 패킷 크기, CPU 사용량을 모니터링하여 DB에 저장 관리한다. <표 3>은 모니터링을 위한 테이블 스키마를 나타낸다.

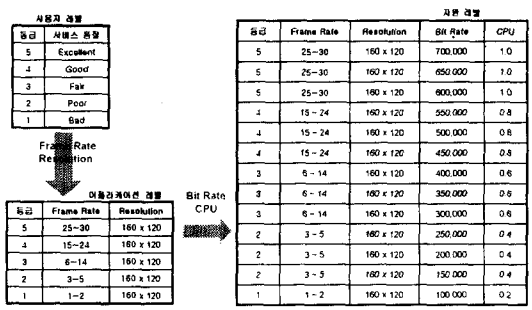
〈표 3〉 DB 스키마

Name	Attribute Name	Type	Sample
T_Monitor	A_Counter	정수(integer)	251
	A_Member	문자열(25)	root@pink
	A_Compression	문자열(10)	jpeg/rtp
	A_Framerate	실수(real)	15.0 f/s
	A_Resolution	문자열(10)	320 x 240
	A_Bitrate	정수(integer)	700,000 bps
	A_BufferSize	정수(integer)	270
	A_PacketSize	정수(integer)	1024
	A_Quality	실수(real)	1.0

5.4.3 QoS MIB

QoS MIB 설계 과정은 하부 네트워크 망의 특성이나 서비스 형태에 따라 가변적이므로 반복된 시험 과정을 통해 설계해야 한다. 이 과정을 통해 얻어진 관

리 정보는 종단간에 QoS 유지를 위해 사용되어진다. QoS MIB 설계과정을 보면 다음과 같은 절차를 통해 구축하였다. 먼저, GUI의 슬라이드 바(Slide Bar)를 통해 사용자가 직접 제어할 수 있도록 다섯 단계의 사용자 레벨로 나누었다.



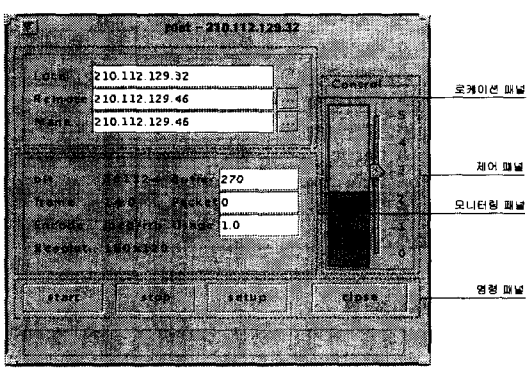
(그림 26) QoS MIB 매핑과정

다음으로 사용자 레벨의 등급에 따라 적절한 디스플레이 프레임율과 해상도를 고려하여 어플리케이션 레벨의 QoS MIB 작성하였다. 해상도는 구현과정에서 고정시켜 QoS MIB을 설계하였다.

마지막으로 어플리케이션 레벨의 QoS 파라미터를 적용하여 만든 테이블에 네트워크 및 시스템 자원에 관련 파라미터인 전송률과 CPU 사용량을 기준으로 세분화하여 QoS MIB 테이블을 얻는다. (그림 26)은 사용자 레벨에서 자원 레벨의 매핑과정을 나타내고 있다.

5.5 사용자 인터페이스(UI) 구현

그래픽 사용자 인터페이스는 (그림 27)과 같다.

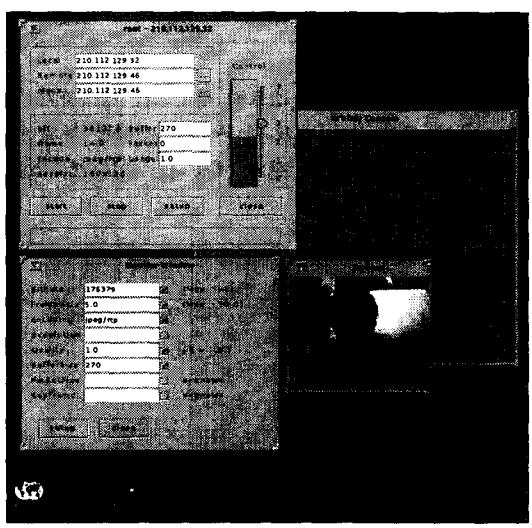


(그림 27) 사용자 인터페이스(GUI)

서비스 접속에 사용하기 위한 각 모듈 객체의 호스트 주소를 지정하기 위한 세 개의 텍스트 필드로 구성된 로케이션 패널(Panel)과 현재 스트림 수신 상태를 표시하는 모니터링 패널과 사용자가 원하는 QoS를 지정하고 현재의 QoS를 나타내는 슬라이드 바와 진행바로 구성된 제어 패널, 마지막으로 스트림 서비스의 시작, 종료, 재설정, 프로그램 닫기 버튼으로 구성된 명령 패널로 구성된다.

5.6 시연 환경

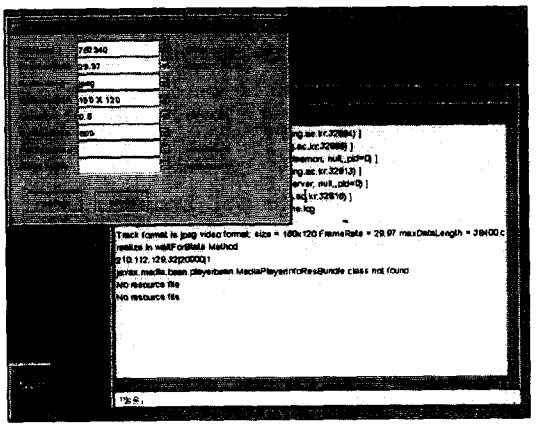
시연 환경은 클라이언트와 서버로 나누어 설명하며, 마지막으로 모니터 객체에 의해 DB에 기록된 모니터링 데이터를 보이고, 협약 기법과 자원 적응 기법의 수행 과정을 통해 QoS 관리가 이루어짐을 보인다. 시연 환경에서 클라이언트는 서버로부터 비디오 스트림 데이터를 받아 디스플레이하고, 서버측은 클라이언트의 요청에 따라 비디오 스트림 데이터를 전송하는 기능을 수행한다. 다음 (그림 28)은 클라이언트측의 전체적인 스트림 수신 화면을 나타내고 있다.



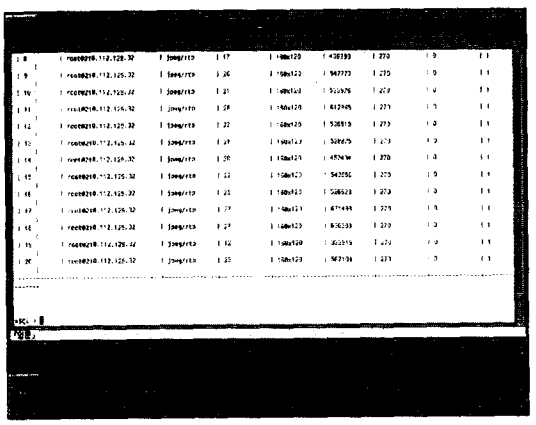
(그림 28) 전체적인 클라이언트측 실행화면

(그림 29)는 서버측의 전체적인 스트림 송신 화면을 나타낸다.

(그림 30)은 스트림 수신측의 모니터 스레드를 통해 DB에 기록되는 T_Monitor 테이블의 내용을 나타낸다.



(그림 29) 전체적인 서버측 실행화면



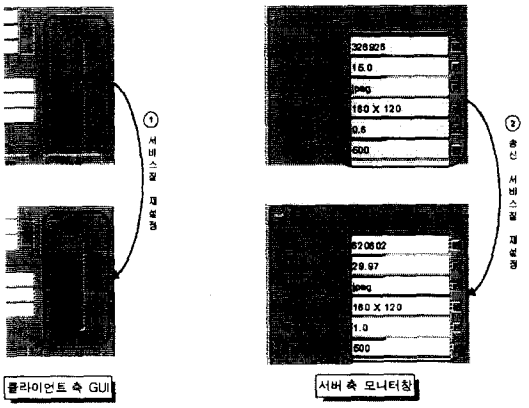
(그림 30) 모니터링 데이터

(그림 31)은 협약 과정의 검증 방법을 나타내며, 협약 과정은 사용자의 요청에 따라 제어 객체가 서비스 질을 재설정하는 과정이다. 다음 요약된 방법을 통해 협약 과정이 수행됨을 검증하였다.

① 사용자가 그래픽 사용자 인터페이스를 이용하여 새로운 서비스 등급을 설정하는 과정을 나타낸다.(3에서 5로 서비스 상향 조정)

② 사용자가 요청하는 QoS를 제공하기 위해 송신측 스트림 객체가 송신 QoS에 관련된 파라미터들을 변화시켜 송신 품질을 재설정하는 과정을 나타낸다.

(그림 32)는 시스템 및 네트워크 자원 변화에 따른 자원 적응 과정의 검증 과정을 나타낸다. 자원 적응

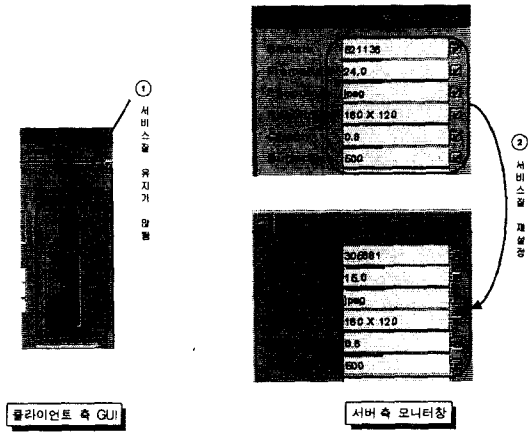


(그림 31) 협약 과정의 검증

과정은 통보 객체, 제어 객체 및 매퍼 객체의 상호작용을 통해 스트림 서비스 과정에서 동적으로 이루어진다. 서비스의 중단을 방지하고 나빠지는 네트워크 상황에 대처하기 위해 서비스 질을 낮추는 과정이다. 이 과정을 통해 절약된 대역폭을 다른 어플리케이션에서 사용하여 자원을 효율적으로 관리한다. 자원 적응 검증 과정을 요약하면 다음과 같다.

① 서비스 수요자와 제공자간에 협약된 서비스 질이 유지되지 못하는 상황이 발생한다.

② 위의 상황에서 송신측 스트림 객체가 트래픽을 줄여 대역폭 절약을 위해 송신 서비스 질에 관련된 파라미터를 자동으로 재설정하는 과정을 나타낸다.



(그림 32) 자원 적응 과정의 검증

6. 결론 및 향후연구

본 논문에서는 객체지향 미들웨어인 CORBA를 이용하여 객체 서비스 형태의 QoS 통합 플랫폼을 제안하였으며, 제안한 플랫폼은 사용자 제어 형태의 협약과 스트림 수신 상황에 동적으로 자원 적응이 가능하도록 사용자 제어 모듈과 QoS 관리 모듈을 설계하였다. 제안한 플랫폼을 설계하기 위해 객체지향 모델링 도구인 UML을 통해 클래스 다이어그램과 시퀀스 다이어그램을 이용하여 객체의 구조, 협약, 자원 적응, 자원 모니터링, 서비스 접속 및 해제의 처리 절차를 모델링하였으며, 데이터 흐름도를 응용하여 객체들간에 메시지와 데이터의 흐름을 설계하였다. 또한 QoS 통합 플랫폼과 스트림 송수신 시스템과 분리된 형태로 설계하여 다양한 응용 스트림 서비스 개발이 용이하고, 플랫폼 독립성과 네트워크 자원의 통합 관리가 가능하도록 설계하고, 재사용 모듈을 사용해 쉽게 멀티미디어 응용 서비스 개발이 가능하도록 일반적인 분산 객체 서비스 형태의 QoS 통합 플랫폼을 제시하였다.

향후 연구과제로는 QoS 파라미터를 더욱 세분화하여 QoS 관리의 정확도를 높이고 서비스 중에 발생하는 예외 상황에 보다 능동적으로 대처할 수 있도록, 예외 처리 클래스와 루틴을 추가하는 연구가 필요하다. 또한 오디오와 비디오 스트림을 동시에 전달하는 다중 스트림 처리 기법과 다중 스트림 처리에 필요한 동기화 기법을 제안하고자 한다. 그리고 스트림 플레이어의 기능을 확장시키고, 다자간 서비스를 위하여 세션 관리(Session Management) 기법을 추가하고자 한다.

참 고 문 헌

[1] Stardust.com, "QoS protocols & architectures. Quality of Service protocols use a variety of complementary mechanisms to enable deterministic end-to-end data delivery," http : //www.qosforum.com

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

[3] H. Schulzrinne, "RTSP Real-Time Streaming Protocol," IETF RFC 1890, April. 1998.

[4] Frank Siqueira, "The Design of a Generic QoS Architecture for Open Systems," Distributed Systems Group, Trinity College Dublin, http : //www.cs.tcd. ie/Frank.Siqueira/PhD-Design/

[5] A. Vogel, G. v. Bochmann, R. Dssouli, J. Gecsei, A. Hafid, B. Kerherve, "On QoS Negotiation in Distributed Multimedia Application," Proc.Protocol for High Speed Networks, April. 1994.

[6] M.Alfano, R. Sigle, "Controlling QoS in a collaborative multimedia environment," Proc. of the 5th IEEE International Symposium on High-Performance Distributed Computing (HPDC-5), Aug 7-9, Syracusae (NY), USA, 1996.

[7] BBN Distributed Systems Projects, "Open Implementation Toolkit for Creating Adaptable Distributed Applications," http://www.dist-systems.bbn.com/project/OIT

[8] OMG, "Control and Management of A/V streams specification," OMG Document telecom /98-10-05, October, 1998.

[9] OMG, CORBAtelecoms : Telecommunications Domain Spec. Version 1.0, Jun. 1998.

[10] Crisina Aurrecoechea, Andrew T.Camphell and Linda Hauw, "A Survey of QoS Architectures," Multimedia Systems Journal, Special Issue on QoS Architecture, 1997.

[11] Witana V. Fry, M. and Antoniades M. "A Software Framework for Application Level QoS Management," To appear in Proc. of the Seventh International Workshop on Quality of Service(IEEE/IFIP IWQoS '99), 1-4th June, 1999.

[12] Michael Fry, Aruna Seneviratne, Andreas Vogel and Varnui Witana, "QoS Management in a World Wide Web Environment Which Supports Continuous Media," Distrib. Syst. Enginerring, Vol.4, pp.38-4, 1997.

[13] Klara Nahrstedt, Hao-hua Chu, Srinivas Narayan, "QoS-aware Resource Management for Distributed

Multimedia Applications," Journal on High-Speed Networking, Special Issue on Multimedia Networking, IOS Press, Vol.8, No.3-4, 1998.

[14] Baochun Li, Klara Nahrstedt, "A Control-based Middleware Framework for Quality of Service Adaptations," in IEEE Journal of Selected Areas in Communication, Special Issue on Service Enabling Platforms, 1999, Vol.17, No.9, September 1999.

[15] I. Busse, B. Deffner, and H. Schulzrinne, "Dynamic QoS control of multimedia applications based on RTP," Computer Communications, Jan. Vol.19, pp. 49-58, 1996.

[16] SunSoft, "Java Media Framework API Guide," http://java.sun.com/products/java-media/jmf/2.0/jmf20-08-apidocs/index.html

전 병 택

e-mail : mate0408@hanmail.net

2000년 원광대학교 컴퓨터공학과 (공학사)

현재 원광대학교 컴퓨터공학과 석사과정

관심분야 : 분산 컴퓨팅, 데이터베이스

김 명 희

e-mail : hee@wonkwang.ac.kr

1993년 원광대학교 컴퓨터공학과 (공학사)

1996년 원광대학교 컴퓨터공학과 (공학석사)

1996년~현재 원광대학교 컴퓨터공학과 박사 과정 중

관심분야 : 멀티미디어 데이터베이스, 분산 실시간 컴퓨팅, 시스템 최적화, 운영체제

주 수 종

e-mail : scjoo@wonkwang.ac.kr

1986년 원광대학교 전자계산공학과 졸업(공학사)

1988년 중앙대학교 컴퓨터공학과 졸업(공학석사)

1992년 중앙대학교 컴퓨터공학과 졸업 (공학박사)

1993년 미국 Univ of Massachusetts at Amherst 전기 및 컴퓨터 공학과 Post-Doc.

1990년~현재 원광대학교 컴퓨터공학과 교수

관심분야 : 분산 실시간 컴퓨팅, 분산객체 모델, 시스템 최적화, 멀티미디어 데이터베이스, 지리정보시스템(GIS)