CORBA 환경에서 실시간 협약 및 적응 제어를 위한 분산 QoS 관리 구조의 설계 및 구현

정회원 이 원 중*, 신 창 선**, 정 창 원**, 주 수 종**

Design and Implementation of Distributed QoS Management Architecture for Real-time Negotiation and Adaptation Control on CORBA Environments

Won-Jung Lee*, Chang-Sun Shin**, Chang-Won Jeong**, Su-Chong Joo** *Regular Members*

요 약

인터넷상에서 멀티미디어 서비스와 스트림 서비스 기대가 증가함에 따라 이를 지원하기 위한 애플리케이션 개발을 많이 요구하게 되었다. 그러나, 기존에 제시된 모델들은 QoS에 관련된 분산 응용 서비스들이 중앙 제어 형태의 통합 모델로 개발됨에 따라 확장이나 재사용을 지원할 수 없는 문제점을 가지고 있다.

따라서, 본 논문에서는 이를 해결하기 위하여 객체지향 미들웨어인 CORBA 상에서 분산 QoS 관리 시스템을 제시하였다. 이는 기존 QoS 제어에서 실시간 협약과 동적 적응 기능뿐만 아니라, 효과적인 QoS 제어 기능도 제공하고 있다. 본 시스템은 QoS의 제어와 관리를 분산된 형태의 시스템으로 설계하였으며, 클라이언트 측에 QoS 제어 관리 모듈(QoS CMM), 서버 측에는 QoS 관리 모듈(QoS MM)을 객체 형태로 각각 구성하였다. 이러한 분산 모듈들은 분산 스트리밍 애플리케이션을 지원하는 동안 분산 QoS 관리를 위해 서로 다른 시스템 상에서 CORBA를 통하여 서로 상호작용 한다. 시스템 설계 단계에서, 스트림 서비스의 QoS 제어를 위해 세부적인 기능과 메소드 호출, 각 모듈의 컴포넌트를 설계하기 위해 UML을 사용했다. 제안된 시스템을 구현하기 위해서 Solaris 2.5/2.7에 CORBA 표준을 따르는 OrbixWeb 3.1c, Java 언어, Java Media Framework API 2.0, Mini-SQL 1.0.16과 SunVideoPlus/SunVideo Capture 보드, Sun Camera와 같은 멀티미디어 장치들을 이용하였다. 마지막으로, 분산 QoS 관리 시스템이 스트림 서비스를 실행하는 동안, 동적으로 클라이언트와 서버 상에서 GUI의 QoS 맵 정보를 근거로, 실시간적으로 협약 및 적응하는 과정을 수치 데이터로 보였다.

ABSTRACT

Nowadays, in accordance with increasing expectations of multimedia stream service on the internet, a lot of distributed applications are being required and developed. But the models of the existing systems have the problems that cannot support the extensibility and the reusability, when the QoS relating functions are being developed as an integrated modules which are suited on the centralized controlled specific-purpose application services. To cope with these problems, it is suggested in this paper to a distributed QoS management system on CORBA, an object-oriented middleware compliance. This systems we suggested can provides not only for efficient control of resources, various service QoS, and QoS control functions as the existing functions, but also QoS control real-time negotiation and dynamic adaptation in addition.

* 군장대학 전자상거래창업과(wjlee@kunjang.ac.kr)
** 원광대학교 컴퓨터공학과(scjoo@wonkwang.ac.kr)
논문번호 : 010021-0305, 접수일자 : 2001년 3월 5일
※ 본 논문은 2001년도 원광대학교 교비지원에 의해서 연구되었습니다.

This system consists of QoS Control Management Module(QoS CMM) in client side and QoS Management Module(QoS MM) in server side, respectively. These distributed modules are interfacing with each other via CORBA on different systems for distributed QoS management while serving distributed streaming applications.

In phase of design of our system, we use UML(Unified Modeling Language) for designing each component in modules, their method calls and various detailed functions for controlling QoS of stream services. For implementation of our system, we used OrbixWeb 3.1c following CORBA specification on Solaris 2.5/2.7, Java language, Java Media Framework API 2.0 beta2, Mini-SQL 1.0.16 and the multimedia equipments, such as SunVideoPlus/SunVideo capture board and Sun Camera. Finally, we showed a numerical data controlled by real-time negotiation and adaptation procedures based on QoS map information to GUIs on client and server dynamically, while our distributed QoS management system is executing a given streaming service.

Ⅰ. 서 론

오늘날 많은 일반 사용자나 응용 서비스 개발자 및 컨텐츠 제공자들은 인터넷상에서 텍스트뿐만 아니라 오디오 및 비디오 데이터를 전송하거나 받을 수 있는 서비스를 요구한다. 이러한 요구는 다양한 멀티미디어 데이터와 스트림 데이터를 전송하는데 필요한 여러 통신 프로토콜[1,2]을 비롯하여, 사용자 요구사항에 따른 서비스 품질(QoS)을 제공하기 위한 여러 가지 서비스 모델과 메커니즘을 제안하고 있다[3]. 한편, 이러한 개방형 추세에 맞추어 시스템 하부구조의 다양성을 감추면서 상위 구조로 하여금 시스템 구조에 무관한 API를 제공함으로서 상위 프로그램과의 이식성을 최대화하는 객체지향 개념과 이를 분산 컴퓨팅 환경에 이용한 분산 객체 기술이 크게 발전하게 되었다. 그리고 최근에는 멀티미디어 데이터를 실시간으로 전송하고 재생하는 멀티미디어 스트리밍 프레임워크가 출현하였다. 이러한 스트리밍이 제대로 이루어지기 위해서는 데이터 전송 중에도 소스(source)와 싱크(sink)가 상호작용을 하며 네트워크 상태에 따라 스트림 전송 속도를 조절해야 하고, 전송 시 손상되거나 손실된 데이터가 있는 경우에도 사용자가 원하는 수준의 서비스 품질을 유지할 수 있어야 한다.

그러나, 현재 개발되고 있는 많은 분산 멀티미디어 응용들은 각 개발회사마다 자체의 스트림 설정이나 제어 메커니즘을 사용하고 있기 때문에 서로 다른 하드웨어 플랫폼, 운영체제, 네트워크 환경에서 상호운용성이 매우 결여되어 있는 실정이다.

따라서, 본 논문에서는 일반적인 QoS 프레임워크에 포함되어야 하는 사항들인 원리, 명세, 메커니즘에 따라 CORBA 기반에서 서로 다른 플랫폼에 독립성을 가지며 다양한 멀티미디어 서비스와 스트림 서비스 사용자의 기대 요구를 보다 만족시키고 이러한 서비스간의 QoS를 보장할 수 있는 분산 QoS 관리 시스템을 제안한다. 제안하고 있는 시스템의 구성은 분산된 형태로 종단간(End-to-End) QoS의 제어와 관리를 할 수 있도록 클라이언트 측에는 QoS 제어 관리를 위한 모듈(QoS CMM)을 서버 측에는 QoS 관리 모듈(QoS MM)로 구성하여 설계하였다. 전체적인 QoS 제어는 클라이언트 측에서 전담하여 다자간 통신의 경우에 서버 측의 부담을 최소화시킬 수 있도록 하였다. 그리고, 각각의 모듈 안에는 이들간의 스트림 서비스와 QoS의 제어 및 관리를 위한 객체들로 구성하였다. 특히, QoS 제어 기법[4,5]중 본 논문에서 제안하는 QoS 제어 기법은 실시간 협약(Real-time Negotiation) 및 동적 적응(Dynamic Adaptation)에 중점을 두었다.

본 논문의 구성은 2장에서 본 연구의 관련 연구로 기존에 제시된 QoS 구조를 비교 및 분석하여 문제점을 제시한다. 3장에서는 QoS에 대한 요구 분석과 본 논문에서 제안하고 있는 분산 QoS 관리 시스템에 대해서 기술한다. 4장에서는 분산 QoS 관리 시스템을 구성하는 각 객체의 정의, 기능, 서비스 수행절차를 UML(Unified Modeling Language)을 이용한 모델링을 보이며, DB 스키마에 대해 나타낸다. 5장에서는 실행 환경에 따른 내용들로 개발 환경, 개발에 필요한 장비, 사용자 인터페이스, 스트림 객체 구현과 실행 환경을 통한 QoS 제어 기법 검증에 대해 기술한다. 마지막으로 6장에서는 결론 및 향후 연구 방향에 대해서 기술한다.

Ⅱ. 관련 연구

최근 몇 년 동안 분산 멀티미디어 응용을 지원하기 위한 노력으로 QoS 범위 내에서 많은 사항들이 고려된 연구가 진행되었고, 이러한 연구 결과로 다양한 QoS 구조가 제안되었다. 따라서, 기존에 제시

된 QoS 구조 중에서 대표적인 구조에 관해 설명하고 구조 및 특징에 대해 기술한다.

2.1 Heidelberg QoS Model

헤이델버그의 IBM 유럽 네트워킹 센터에서 네트워크와 종단 시스템(End-system)들의 보정을 제공하기 위한 QoS 모델[6]을 제시하였다. 설계 목적은 멀티캐스트 그룹에서 개별적인 수신자들이 요구하는 이질적인(heterogeneous) QoS를 처리하고 플로우 필터링(flow filtering)과 매체 스켈링 기술(media scaling technique)을 경유하여 적응적인 QoS를 제공하는데 있다. 통신 구조는 연속적인 미디어 전송 시스템(HeiTS/TP)을 포함하고, 이는 매체 스켈링과 QoS 매핑을 제공한다. 기본 전송은 ST-II 상호운용 계층에서 통계적인 서비스 레벨과 보정을 지원하며, QoS 기반 라우팅과 QoS 필터링 기능을 제공한다. 이 모델에서 중심 부분은 종점간 보정을 제공하는 HieRAT(Resource Administration Technique)이며, 이는 QoS 관리 스킴으로 QoS 협약, QoS 계산, QoS 승인 제어, QoS 시행, QoS 자원 스케줄링으로 구성된다. 다음의 (그림 1)은 Heidelberg QoS 모델을 나타내고 있다.

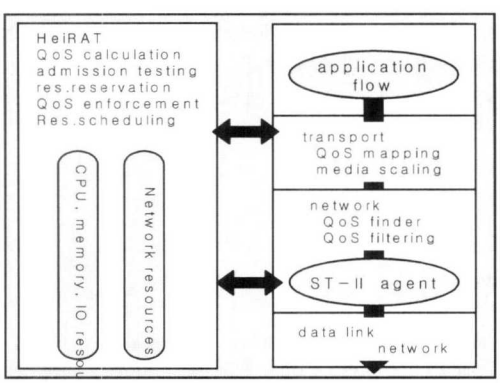


그림 1. Heidelberg QoS Model

2.2 OMEGA

펜실베니아 대학에서 개발된 OMEGA 구조[7]는 애플리케이션 QoS 요구사항간의 관계를 잘 나타내고 있으며, 지역 시스템의 가용능력과 필요한 요구사항에 적합한 글로벌 자원 관리를 효과적으로 할 수 있다. 또한, 실시간 QoS 보정 제공과 자원 예약 및 종점간 자원 관리 기능을 제공한다. OMEGA 구조에서 QoS broker 기능은 통합 QoS 전송과 QoS 협약 및 재협약을 담당한다. 다음의 (그림 2)는 OMEGA 구조를 나타내고 있다.

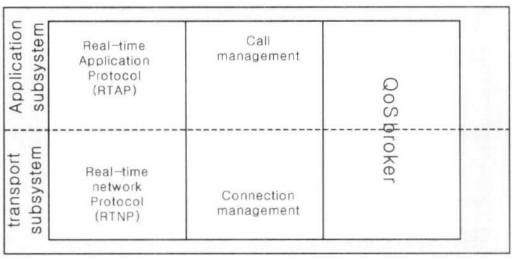


그림 2. OMEGA

2.3 OSI QoS Framework

OSI에서는 QoS 지원에 중점을 두고 있는 OSI QoS 프레임워크를 제시하고 있다[8]. 여기에서는 QoS를 위한 개념과 용어를 정의하고 있고, QoS와 관련된 객체들과 이들간의 상호 작용은 QoS 특성의 집합을 정의한 것을 통해서 기술된다. OSI QoS 프레임워크의 중요한 특징은 QoS 요구사항, QoS 특징, QoS 범주, QoS 관리 기능을 포함하고 있다. 다음의 (그림 3)은 OSI QoS 프레임워크를 나타내고 있다.

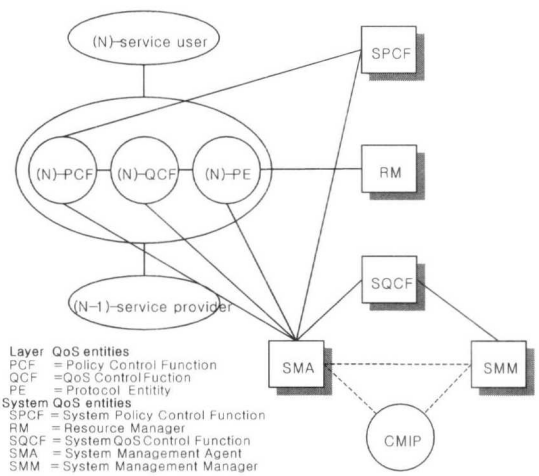


그림 3. OSI QoS 프레임워크

2.4 Tenet Architecture

Tenet 구조[9]는 캘리포니아 대학의 Tenet 연구 그룹에서 개발하였으며, 이는 광역 ATM 네트워크 상에서 실험적인 프로토콜의 집합으로 구성되어 있다. Tenet 구조에서 중요한 구성 요소는 RCAP (Real Time Channel Administration Protocol), RTIP(Real Time Internet Protocol), CMIP(Continuous Media Transport Protocol)로 구성된다. 다음의 (그림 4)는 Tenet 구조를 나타내고 있다.

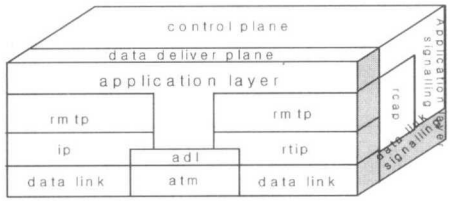


그림 4. Tenet 구조

2.5 MASI End-to-End Model

MASI 구조는 종점간 QoS를 지원하는 멀티미디어 통신을 위한 구조로 MASI 연구소의 CESAME 프로젝트와 Pierre et Marie Curie 대학에서 개발하였다[10]. MASI 구조는 ATM 기반의 네트워크 상에서 분산 멀티미디어 애플리케이션 동작에 필요한 QoS 요구 구현과 명세를 위한 QoS 프레임워크를 제시하고 있다. (그림 5)는 MASI 구조를 나타내고 있다.

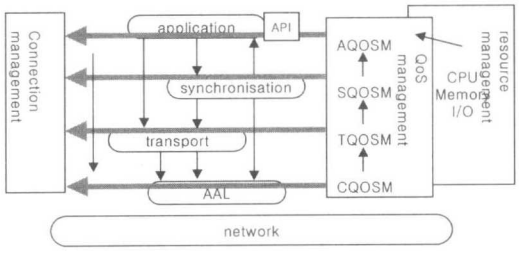


그림 5. MASI 구조

2.6 QoS 구조 비교

위의 관련 연구에서는 기존에 제시된 QoS 구조 중에서 대표적으로 몇 가지를 선택하여 이들 구조의 특징 및 구조에 대해서 기술하였다. 여기에서는 제시한 구조들에 대한 비교로써 <표 1>을 통해 전체적으로 나타낸다.

2.7 문제점

앞에서 언급한 관련연구에 의해서 제시된 모든 QoS 구조는 애플리케이션 레벨의 QoS 요구사항에 기초를 둔 QoS 명세를 고려하고 있다.

그러나, QoS를 위한 프레임워크들은 분산 멀티미디어 응용을 위해 개별적인 스트림 설정이나 제어 메커니즘을 사용하고 있으며, 서로 다른 플랫폼, 운영체제, 통신 프로토콜, 네트워크 환경 등의 각 범주에 제한된 연구가 진행되어 인터넷과 같은 분산 망에서는 상호운용성의 문제점이 있다. 또한, 종단 사용자(End-user)의 QoS 정책 지원에 있어서 사용자 요구를 배제한 멀티미디어 트래픽 제어 및 관리 목적의 인터페이스 제공이나 서비스 수준 계약(Service Level Agreement)시 QoS 분산 제어 및 유지 보수 등의 예측 메커니즘의 부족과 이질적이고 다양한 시스템 및 네트워크를 고려한 통합 QoS 프레임워크의 부족과 같은 문제점을 가지고 있다. 그리고, 이러한 연구의 대부분이 특정 프로토콜이나

표 1. 제시된 QoS 구조에 대한 특징 비교

QoS Model	Qos Provision QoS Control								QoS Management	
QoS Mechanism	QoS Mapping	Adm Control/ Resource allocation	E2Ea Coordination	Flow Scheduling	Flow Shaping	Flow Control	QoS Filtering	Flow Synchronization	Monitoring/ Alerts	QoS Maintenance
XRM	EN	EN	(E)N	(E)N	-	N	-	-	N	-
QoS-A	EN	E(N)	EN	E(N)	E	(E)	(E)N	E	EAD	ENRS
ISO	(E)(N)	EN	EN	EN	E(N)	(N)	N	-	EN	EN
Heidelberg	(E)(N)	EN	EN	E(N)	(E)	(N)	N	-	ED	ERS
TINA	(E)	(N)	N	-	-	-	-	(N)	(N)	-
IETF	EN	-	E	-	-	-	-	-	EN	ENR
Tenet	E(N)	E(N)	E	E	-	-	-	E	E	E
MASI	E(N)	E(N)	E	E	-	-	-	E	E	E
OMEGA	E(N)	E(N)	E(N)	E(N)	E	E	-	-	E	ER
WashU	E	E		E	E	-	-	-	-	ER

- : 다루지 않음, S : 세부적인 QoS 스케링 처리, E/N : 종점 시스템/네트워크에서 세부적으로 다룸
D : 세부적인 QoS Degradation 처리, (E)/(N) : 종점 시스템/네트워크에서만 적용, (D) : QoS Degradation만 적용
R : 세부적인 QoS 재협약 처리, A : 세부적인 QoS 이용가능, (R) : QoS 재협약만 처리

특정 네트워크 환경에 적용되므로 QoS 기능이 매우 제한적이며, 확장성이나 재사용성의 문제점을 가지고 있고, QoS 제어와 관리 메커니즘에 있어서도 실시간 특성을 고려한 제어와 관리 기능이 매우 부족한 실정이다.

따라서, 본 논문에서는 분산 객체 기반의 분산 멀티미디어 응용 프로그램을 지원할 수 있는 CORBA 기반의 프레임워크를 제시하고, 이를 기반으로 스트림 서비스를 위한 객체들의 기능을 정립하고 설계한다. 그리고, 실시간 멀티미디어 서비스를 위해 다양한 멀티미디어 정보를 처리하고, QoS 제어 및 관리를 효과적으로 할 수 있는 제어 관리 객체에 중점을 둔 분산 QoS 관리 구조를 제시한다.

Ⅲ. 분산 QoS 관리 구조

3.1 고려사항

본 논문에서 제안하고 있는 분산 QoS 관리 구조는 일반적인 QoS 프레임워크를 위한 사항들인 QoS 원리, QoS 명세, QoS 메커니즘에 기초를 두었다. 그리고 애플리케이션 레벨의 QoS 관점과 QoS 명세에서 관리 정책에 의한 QoS 제어 메커니즘에 중점을 두어 실시간 협약 및 동적 적응의 제어가 가능하도록 하여 CORBA 기반에서 분산 제어 형태로 다양한 멀티미디어 서비스와 스트림 서비스의 서비스 품질을 보장하고, 제어 및 관리가 용이하다. 여기에서 중점을 두고 있는 QoS 제어 메커니즘으로는 QoS 모니터링, QoS 통보, QoS 협약, QoS 적응이 있다. 또한, 다음과 같은 사항들을 중점적으로 고려하였다.

- 플랫폼에 독립적이며, 이질적인 시스템간의 상호운용성 제공
- 자원들에 대한 효율적인 관리와 실시간 특성을 지원
- 확장성과 재사용이 가능한 QoS 제어 및 관리 모듈 제공

3.2 구조

분산 QoS 관리 구조를 이루는 객체 모듈은 클라이언트 측의 QoS 제어 관리 모듈(QoS CMM : QoS Control Management Module)과 서버 측의 QoS 관리 모듈(QoS MM : QoS Management Module)로 구성되며, 각각의 모듈 내부는 종단간 스트림 서비스의 QoS를 보장하기 위한 객체들로 구성되어 있다. 이들간의 스트림 흐름은 IP기반에서 RTP 패킷을 송수신할 수 있는 스트림 송수신 객체에서 처리하고, 제어 흐름은 CORBA 기반의 ORB를 통해서 이루어진다. 다음 (그림 6)은 제안하고 있는 분산 QoS 관리 구조를 나타낸다.

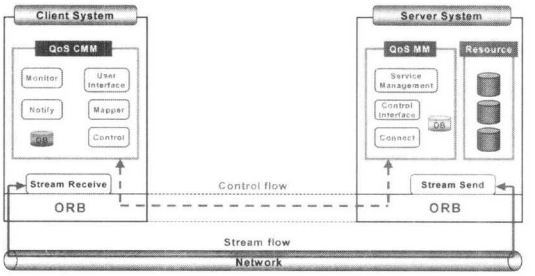


그림 6. 분산 QoS 관리 구조

Ⅳ. 분산 QoS 관리 구조 설계

본 장에서는 먼저 분산 QoS 관리 구조를 이루고 있는 객체에 대한 기능을 구체적으로 정의하여 클래스 다이어그램으로 나타내고, QoS 제어 기법의 수행과정을 통해 객체들간의 관계 및 데이터 흐름을 보이며, 객체들간의 메시지 흐름을 통해 분산 QoS 관리 구조의 전반적인 수행절차와 각각의 서비스 수행 절차를 보이도록 한다. 이를 위한 모델링 도구로는 UML을 이용하여 QoS 제어 관리 모듈과 QoS 관리 모듈은 클래스 다이어그램으로 나타내고, 제어기법 중 접속과 실시간 협약 및 동적 적응, 자원 모니터링, 접속 해제 절차는 시퀀스 다이어그램을 통해서 나타낸다.

4.1 QoS 제어 관리 모듈

QoS 제어 관리 모듈은 사용자 정보를 나타내기 위한 클래스와 클라이언트와 서버간의 실시간 협약과 동적 적응을 위한 서비스 상태 정보를 전달하기 위해 사용되는 클래스로 구성된다. 다음의 (그림 7)은 QoS 제어 관리 모듈의 스테레오 타입을 적용한 클래스 다이어그램을 나타내고 있다.

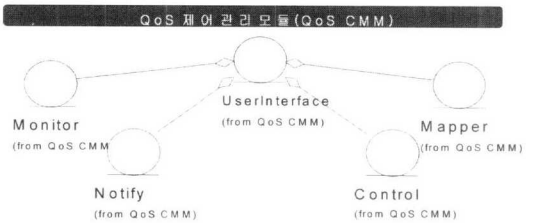


그림 7. QoS 제어 관리 모듈의 클래스 다이어그램

4.1.1 사용자 인터페이스 객체

사용자 인터페이스 객체는 주로 스트림 응용 프로그램의 개발에 사용할 수 있는 인터페이스 제공을 목적으로 함으로써 QoS 제어 관리 모듈을 구성하고 있는 모니터 객체, 매퍼 객체, 제어 객체, 통보 객체에 대한 관리를 담당한다. 다음 (그림 8)은 사용자 인터페이스 객체의 클래스 다이어그램을 나타낸다.

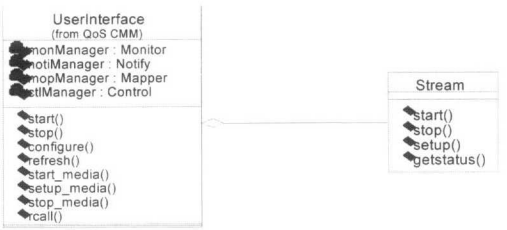


그림 8. 사용자 인터페이스 객체의 클래스 다이어그램

4.1.2 모니터 객체

모니터 객체는 서버 측의 스트림 객체로부터 송신되는 스트림 데이터의 수신 상태와 스트림 송신 시스템의 자원 사용량을 DB에 기록하는 모니터 스레드와 모니터링 주기를 나타내는 속성으로 구성된다. 또한, 모니터링에 관련된 메소드는 스레드의 제어 기능을 수행한다. 다음 (그림 9)는 클래스 다이어그램을 나타내고 있다.

그림 9. 모니터 객체의 클래스 다이어그램

4.1.3 통보 객체

통보 객체는 모니터 객체가 기록한 모니터링 데이터와 정보를 분석하여 협약된 서비스에 맞게 QoS가 유지되는지를 감시한다.

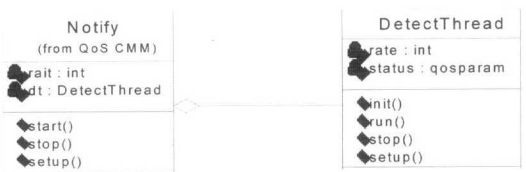


그림 10. 통보 객체의 클래스 다이어그램

또한, 모니터 객체와 같이 스레드를 통해 수행하도록 스레드 객체 참조자를 유지한다. 위 (그림 10)은 통보 객체의 클래스 다이어그램을 나타내고 있다.

4.1.4 매퍼 객체

매퍼 객체는 접속 가능한 서버 시스템 목록인 시스템 카탈로그 정보를 관리하고, 클라이언트가 서비스 요청을 할 경우에 서버 측과의 연결 설정 및 해제에 관련된 정보를 유지하며, QoS MIB를 관리하는 역할을 담당한다. 다음의 (그림 11)은 매퍼 객체의 클래스 다이어그램을 나타내고 있다.

그림 11. 매퍼 객체의 클래스 다이어그램

4.1.5 제어 객체

제어 객체는 클라이언트 측의 현재 시스템 정보와 사용자의 서비스 요청 정보를 바탕으로 실시간 협약 및 동적 적응 과정을 처리하는 객체로 매퍼 객체의 QoS MIB를 기초로 클라이언트와 서버간의 QoS를 유지하는 기능을 제공한다. 다음의 (그림 12)는 제어 객체의 클래스 다이어그램을 나타내고 있다.

그림 12. 제어 객체의 클래스 다이어그램

4.2 QoS 관리 모듈

QoS 관리 모듈은 클라이언트 측에서 요청한 서비스 레벨에 맞게 송신 시스템을 재구성하거나 요청된 서비스가 협약된 사항에 위배될 경우 이를 다시 재조정하여 서비스를 송신할 수 있도록 한다. 그리고 자원들에 대한 관리와 제어를 담당한다. 다음의 (그림 13)은 QoS 관리 모듈의 스테레오 타입을 적용한 클래스 다이어그램을 나타낸다.

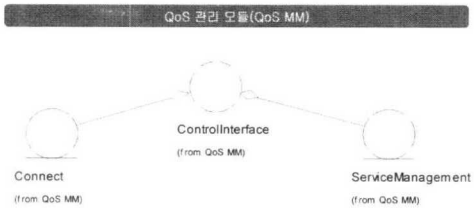


그림 13. QoS 관리 모듈의 클래스 다이어그램

4.2.1 접속 객체

접속 객체는 클라이언트 측에서 연결 접속을 요청할 때 두 시스템간의 연결 설정 및 해제 기능을 수행하고, 매퍼 객체가 제공하는 시스템 카탈로그를 통해서 서비스를 설정한다. 다음의 (그림 14)는 접속 객체의 클래스 다이어그램을 나타내고 있다.

그림 14. 접속 객체의 클래스 다이어그램

4.2.2 서비스 관리 객체

서비스 관리 객체는 클라이언트 측에서 요청하는 서비스 레벨에 맞게 스트림 송신 객체를 재구성하도록 하고, 서버 측에 있는 자원들을 관리하여 클라이언트가 요청하는 서비스가 정확하게 송신될 수 있도록 하는 역할을 한다. 다음 (그림 15)은 서비스 관리 객체의 클래스 다이어그램을 나타내고 있다.

그림 15. 서비스 관리 객체의 클래스 다이어그램

4.2.3 제어 인터페이스 객체

제어 인터페이스 객체는 QoS 관리 모듈내의 객체들은 통합 관리할 수 있는 인터페이스를 제공한다. 그리고, 클라이언트 측에서 전달되는 제어 정보에 의해 현재 시스템 상황에 맞는 서비스 레벨 또는 사용자가 요청한 서비스 레벨에 맞게 QoS 레벨을 재조정하는 역할을 담당한다. 다음의 (그림 16)은 제어 인터페이스 객체의 클래스 다이어그램을 나타내고 있다.

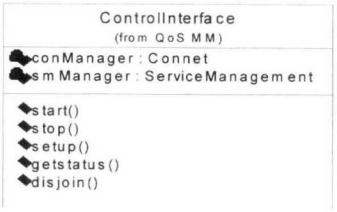


그림 16. 제어 인터페이스 객체의 클래스 다이어그램

4.3 객체 기능 설계

객체간의 관계를 중심으로 분산 QoS 관리 구조의 서비스 수행 절차와 본 논문에서 제안된 QoS 제어 기법을 객체간의 데이터 흐름을 통해 설명하고, 이들간의 구체적인 기능 및 역할에 대해서 기술한다.

4.3.1 분산 QoS 관리 구조의 수행 절차

다음의 (그림 17)은 분산 QoS 관리 구조에 있어서 전반적인 수행 절차를 ETD로 표현하였다.

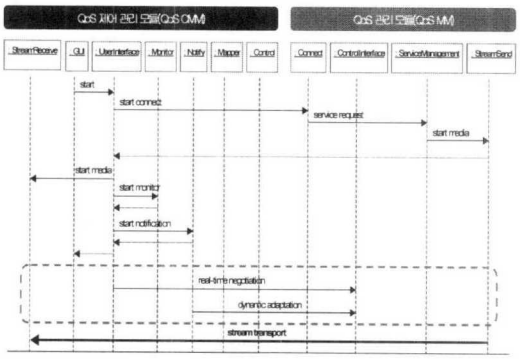


그림 17. 분산 QoS 관리 구조의 수행 절차

4.4 객체간 시나리오 설계

본 절에서는 제안하고 있는 분산 QoS 관리 구조의 수행 절차를 서비스 접속, 접속 해제, 실시간 협약, 동적 적응, 자원 모니터링 수행 절차로 나누어 기술한다. 이를 위한 표기법으로는 UML의 시퀀스 다이어그램을 이용해서 나타낸다.

4.4.1 접속 수행 절차

서비스 요청의 클라이언트 측과 요청된 서비스를 제공하기 위한 서버 측의 스트림 서비스 접속 과정은 초기에 서비스를 요청하는 사용자가 그래픽 사용자 인터페이스(GUI)를 이용하여 QoS 제어 관리 모듈의 사용자 인터페이스 객체에 있는 시작(start)

오퍼레이션을 통해 QoS 관리 모듈의 접속 객체에게 접속을 요청한다. 이때 접속이 성공되면 QoS 관리 모듈의 스트림 송신 객체는 스트림 데이터의 전송을 시작하고, QoS 제어 관리 모듈의 스트림 수신 객체는 전송되는 스트림 서비스를 수신할 수 있도록 하며, 스트림 수신 상태를 모니터링하고 감시할 수 있도록 모니터 객체와 통보 객체를 구동시킴으로써 서비스의 연결 접속이 시작된다.

다음의 (그림 18)는 종단간 스트림 서비스 접속 수행 절차의 시퀀스 다이어그램을 나타내고 있다.

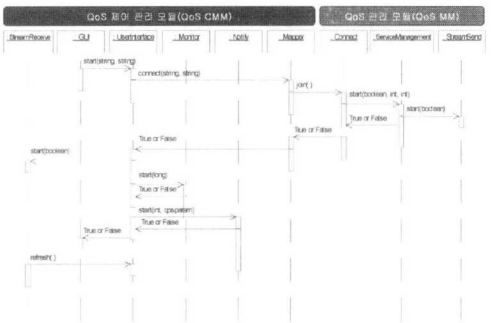


그림 18. 스트림 서비스 접속 수행 절차 시퀀스 다이어그램

4.4.2 실시간 협약 수행 절차

종단간 스트림 서비스의 실시간 협약 수행 절차는 초기 서비스를 요청하는 클라이언트 측의 QoS 제어 관리 모듈의 사용자 인터페이스 객체를 통해 현재 사용자가 요구하는 QoS 정보를 제어 객체에게 전달한다. 정보를 전달받은 제어 객체는 매퍼 객체에게 QoS MIB를 요청한 후, 제공된 정보를 바탕으로 협약 알고리즘을 수행한다. 그리고, QoS 관리 모듈의 제어 인터페이스 객체에게 협약된 결과인 QoS 파라미터 정보를 전달한다. 정보를 넘겨받은 제어 인터페이스 객체는 요청된 서비스 레벨에 맞게 서비스 레벨을 재구성하고, 서비스 관리 객체에게 재구성된 서비스 레벨에 맞추어 스트림 서비스를 송신할 수 있도록 요청한다. 서비스 관리 객체는 스트림 송신 객체를 서비스 레벨에 맞추어 재구성한 후 클라이언트 측으로 서비스를 송신한다. 다음의 (그림 19)은 실시간 협약 수행 절차의 시퀀스 다이어그램을 나타내고 있다.

4.4.3 동적 적응 수행 절차

동적 적응 수행 절차는 QoS 제어 관리 모듈의 통보 객체가 서비스 상황 중에 클라이언트 측과 서버 측의 협약된 QoS 레벨로 서비스를 받지 못하는

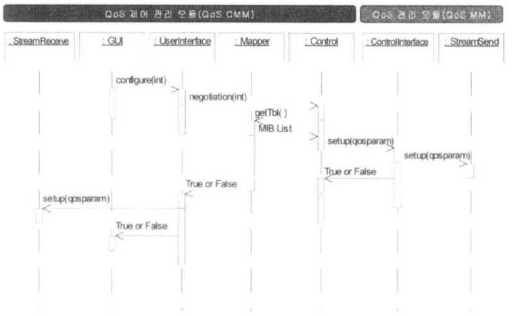


그림 19. 실시간 협약 수행 절차의 시퀀스 다이어그램

경우를 발견함으로써 시작된다. 통보 객체는 현재의 서비스 상태 정보와 같이 적응 알고리즘을 수행하도록 제어 객체에게 요청한다. 요청을 받은 제어 객체는 현재의 서비스 상태 정보를 참조하여 QoS 관리 모듈의 제어 인터페이스 객체에게 현재 서비스 상태에 맞게 서비스 레벨의 재 설정을 요구한다. 제어 인터페이스 객체는 서비스 레벨을 재 설정한 후 설정된 서비스 레벨에 맞추어 서비스를 송신할 수 있도록 서비스 관리 객체에게 요청한다. 요청을 받은 서비스 관리 객체에 의해서 스트림 송신 객체의 재 설정을 통해 클라이언트 측과 서버 측의 동적 적응이 수행된다. 다음의 (그림 20)은 동적 적응 수행 절차의 시퀀스 다이어그램을 나타내고 있다.

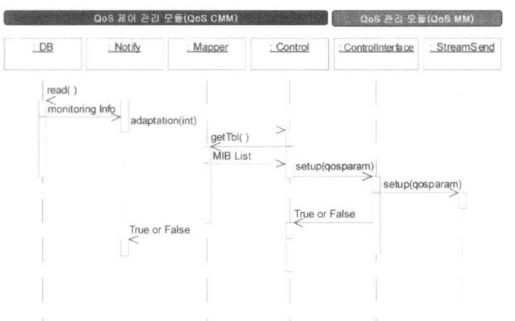


그림 20. 동적 적응 수행 절차의 시퀀스 다이어그램

4.4.4 자원 모니터링 수행 절차

자원 모니터링 수행 절차는 모니터 객체가 사용자 인터페이스 객체에게 스트림 수신 상태 정보를 요청하고 사용자 인터페이스 객체는 스트림 수신 객체로부터 수신 상태 정보를 얻는다. 이렇게 얻은 정보는 DB 테이블에 기록되며, 이러한 과정을 반복 처리하여 자원 모니터링 과정이 수행된다. 다음의 (그림 21)은 자원 모니터링 수행 절차를 나타내고 있다.

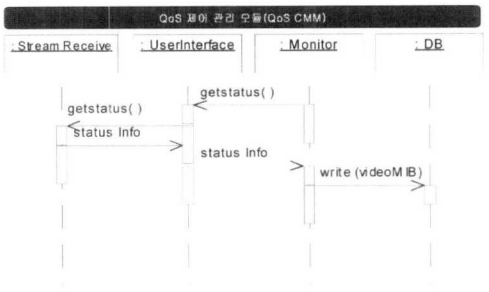


그림 21. 자원 모니터링 수행 절차의 시퀀스 다이어그램

4.4.5 접속 해제 수행 절차

접속 해제 수행 절차는 클라이언트 측의 사용자가 그래픽 사용자 인터페이스 화면에 있는 정지(stop) 버튼을 누름으로써 시작된다. 사용자 인터페이스 객체는 매퍼 객체에게 접속 해제를 요청하고, 매퍼 객체는 QoS 관리 모듈의 접속 객체에게 연결 해제를 요청한다. 연결 해제 요청 후 서비스 관리 객체를 통해 스트림 객체를 종료시켜 클라이언트와 서버간의 스트림 서비스의 연결 접속을 해제한다.

다음의 (그림 22)는 접속 해제 수행 절차의 시퀀스 다이어그램을 나타내고 있다.

그림 22. 접속 해제 수행 절차의 시퀀스 다이어그램

4.5 데이터베이스 스키마 설계

모니터링하기 위해 사용되는 데이터베이스 스키마는 클라이언트 측의 스트림 객체로부터 스트림 데이터 수신 상태를 기록하고 관리한다. 모니터링 데이터는 전송률, 프레임 전송률, 엔코더 형식, 해상도, 버퍼 크기, 패킷 크기, CPU 사용량이며, 이러한 데이터는 스트림 서비스 동안 주기적으로 얻어진다. 다음 <표 2>는 클라이언트 측의 모니터링을 위한 데이터베이스 스키마를 나타내고 있다. 이는 서버 측의 데이터베이스 스키마와 같다. 단, 서버 측의 모니터링 데이터는 클라이언트 측에서 모니터링된 협약된 데이터와 적응하기 위한 데이터만이 기록되고 관리된다.

표 2. 클라이언트 측의 모니터링을 위한 데이터베이스 테이블 스키마

Table Name	Attribute Name	Attribute Type	Sample
C_Monitor	C_Counter	정수(integer)	250
	C_Member	문자열(25)	210.112.129.32
	C_Framerate	실수(real)	26.0 fps
	C_Resolution	문자열(10)	160×120
	C_Bitrate	정수(integer)	692,000 bps
	C_Compression	문자열(10)	jpeg/rtp
	C_BufferSize	정수(integer)	270
	C_PackSize	정수(integer)	1024
	C_Quality	실수(real)	1.0

V. 분산 QoS 관리 구조 시스템 구현

본 장에서는 앞서 기술한 분산 QoS 관리 구조에 대한 설계를 바탕으로 이를 구현한 내용을 기술한다. 구현에 있어서 사용하게된 시스템, 멀티미디어 장비, 개발 언어, ORB, 스트림 개발의 API, 플랫폼 환경 등에 관한 사항들을 설명하고, 설계된 객체를 CORBA 상에서 객체화하기 위한 IDL 설계와 스트림 객체 구현에 관련된 사항에 대해서 기술한다. 그리고, 끝으로 실행 화면을 제시하여 실시간 협약 및 동적 적응에 대해서 보인다.

5.1 개발 환경

본 절에서는 분산 QoS 관리 구조를 구현하는데 있어서 사용되는 각각의 요소들에 대한 사항들을 세부적으로 설명한다. 구현된 시스템은 클라이언트 측에 QoS 제어 관리 모듈을 등록하고, 서버 측에는 QoS 관리 모듈을 등록하여 클라이언트와 서버간의 스트림 송수신이 가능하도록 구현하였다. 본 논문에서 제안하고 있는 분산 QoS 관리 구조의 프로토타입 환경은 다음의 (그림 23)와 같다.

5.2 IDL 정의

앞서 제시한 분산 QoS 관리 구조를 구성하고 있는 객체들을 CORBA 객체로 객체화하기 위해 객체 표준을 정의하고 있는 OMG의 IDL로 정의하였다. 그리고 정의한 IDL을 중심으로 구현될 각각의 객체를 구성하는 속성과 오퍼레이션에 대해서 설명한다.

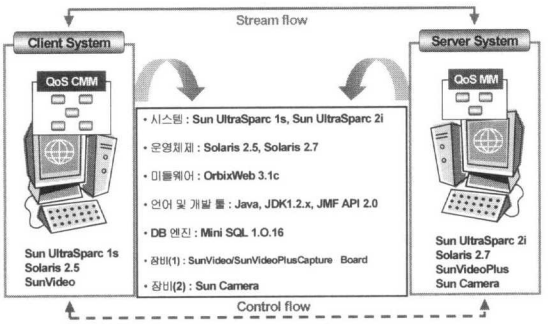


그림 23. 분산 QoS 관리 구조의 개발 환경

먼저 두 모듈 객체간에 공통적으로 사용되는 자료 구조에 대해서 설명하고, QoS 제어 관리 모듈의 객체에 대한 IDL 정의와 QoS 관리 모듈의 객체 IDL 정의에 대해서 설명한다.

5.2.1 자료 구조 정의

다음의 (그림 24)는 두 모듈 객체간에 공통적으로 사용되는 자료 구조에 대한 정의를 나타낸다.

```
struct member {              //사용자 정보 구조
   string      name ;
   string      ip;
   string      hostname; };
struct videoMIB {            //Video QoS MIB 구조
   long        level;
   float       framerate;
   string      encoder;
   string      resoultion;
   float       cpu;
   long        buffersize;
   long        packetsize; };
typedef sequence<member> catalogList;
  //카탈로그 목록, QoS MIB 목록
typedef sequence<videMIB> mibList;
```

그림 24. 자료 구조 정의

먼저 member 구조체는 종단간 스트림 서비스 연결 과정에서 전달되는 접속 시스템에 대한 사용자 정보로 사용자 이름, 접속 시스템의 IP 주소와 호스트 이름으로 구성된다. videoMIB 구조체는 스트림 서비스를 제어하는데 사용될 정보로 구성은 QoS, 프레임 전송률, 인코딩 포맷, 해상도, CPU 사용량, 버퍼 크기, 패킷 크기로 구성된다. 그리고, qosparam 구조체는 종단간 협약이나 적응 알고리즘 수행 과정에서 스트림 서비스의 상태 정보를 나타내기 위한 파라미터로 사용되며, 구성은 프레임 전송률, 해상도, CPU 사용량으로 구성된다. 마지막으로 카탈로그 목록인 catalogList와 QoS MIB목록인 mibList는 두 가지 목록 정보를 가진다.

5.2.2 QoS 제어 관리 모듈의 IDL 정의

QoS 제어 관리 모듈을 구성하고 있는 사용자 인터페이스 객체, 모니터 객체, 통보 객체, 매퍼 객체, 제어 객체의 IDL 정의에 대해서 설명한다.

■ 사용자 인터페이스 객체

다음의 (그림 25)는 사용자 인터페이스 객체의 IDL을 나타낸다.

```
interface UserInterface {
   attribute Monitor          monManger;
   attribute Notifiy          notiManager;
   attribute Mapper           mapManager;
   attribute Control          ctlManager;
   boolean    start(in string local, in string remote, in string
                    dbport);
   boolean stop();
   boolean configure(in long qoslevel);
   boolean refresh();
   boolean start_media(in qosparam param);
   boolean stop_media();
   boolean stepup_media(in qosparam param);
   boolean rcall(in string local, in string remote); };
```

그림 25. 사용자 인터페이스 객체의 IDL

먼저 애트리뷰트에서 QoS 제어 관리 모듈을 구성하는 모니터 객체, 통보 객체, 매퍼 객체, 제어 객체에 대한 객체 참조자를 관리하기 위해서 네 가지의 애트리뷰트를 정의하였다. 그리고, 서비스 시작을 위해 start() 오퍼레이션, 협약을 위해 사용되는 configure() 오퍼레이션, GUI 화면의 모니터링 시작을 위한 refresh() 오퍼레이션, 송신 시작을 요청하는 start_media() 오퍼레이션, 스트림 송신을 재설정하기 위한 setup_media() 오퍼레이션, 스트림 송신 정지를 위한 stop_media() 오퍼레이션을 제공하며, QoS 관리 모듈 객체들의 초기화를 요청하는 rcall() 오퍼레이션을 제공한다.

■ 모니터 객체

다음의 (그림 26)은 모니터 객체의 IDL이다.

```
interface Monitor {
   attribute MonitorThread mt;
   boolean start(in long rate);
   boolean stop();
   boolean setup(in long rate);
   boolean initial(in UserINterface ui); };
```

그림 26. 모니터 객체의 IDL

모니터 객체는 모니터 스레드의 객체 참조자를 위해 애트리뷰트로 모니터 스레드를 두고, 스트림 데이터의 수신 상태를 기록하기 위한 start() 오퍼레이션과 해제를 위한 stop() 오퍼레이션을 제공하며, 데이터를 DB에 저장하기 위해 setup() 오퍼레이션을 제공한다. 그리고, 모니터 객체를 초기화하기 위

해 initial() 오퍼레이션을 제공한다.

■ 통보 객체

다음의 (그림 27)은 통보 객체의 IDL이다.

```
interface Notity {
 attribute DetectThread dt;
 boolean start(in long rate, in qosparam param);
 boolean stop();
 boolean setup(in long rate, in qosparam param); };
```

그림 27. 통보 객체의 IDL

통보 객체도 모니터 객체와 같이 스레드 객체의 참조자를 위해 애트리뷰트로 검출 스레드를 두고, 위반 사항의 검출을 위해 검출 시작과 종료기능을 담당하는 start() 오퍼레이션과 stop() 오퍼레이션을 제공하며, 검출된 데이터를 저장하기 위해 setup() 오퍼레이션을 제공한다.

■ 매퍼 객체

다음의 (그림 28)은 매퍼 객체의 IDL이다.

```
interface Mapper {
 attribute catalogList serverTbl;
 attribute mibList videoTbl;
 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 UserInterface ui, in string remote); };
```

그림 28. 매퍼 객체의 IDL

매퍼 객체는 앞서 나타낸 자료구조에서 설명한 것처럼 접속 가능한 시스템 목록을 관리하기 위해 애트리뷰트 catalogList와 협약과 적응 기법에 사용되는 QoS MIB 목록을 관리하기 위한 애트리뷰트 mibList를 가진다. 그리고, start() 오퍼레이션은 매퍼 객체의 활성화를 위해서 제공하고, stop() 오퍼레이션은 비활성화를 위해서 제공된다. QoS 제어 관리 모듈의 사용자 인터페이스 객체의 요청에 따라 세션에 참여 또는 탈퇴시키기 위한 connect() 오퍼레이션과 disconnect() 오퍼레이션을 제공하고, catalog 목록을 새로 갱신하기 위한 register() 오퍼레이션과 unregister() 오퍼레이션을 제공한다.

또한, 제어 객체에게 QoS MIB 목록을 제공하기 위한 getVideoTbl() 오퍼레이션을 제공하고, 접속 객체에게 catalog 목록을 제공하기 위해 getServer Tbl() 오퍼레이션을 제공하며, 매퍼 객체를 초기화하기 위해 initial() 오퍼레이션을 제공한다.

■ 제어 객체

다음의 (그림 29)는 제어 객체의 IDL이다.

```
interface Control {
 boolean start();
 boolean stop();
 boolean negotiation(in long qoslevel);
 boolean adaptation(in long qoslevel, in qosparam param);
 boolean initial(in UserInterface ui, in string remote);};
```

그림 29. 제어 객체의 IDL

제어 객체는 start() 오퍼레이션과 stop() 오퍼레이션을 제공하여 객체를 활성화 또는 비활성화 시키는 기능을 담당하게 하고, 협약 기법을 처리하기 위해 negotiation() 오퍼레이션과 적응 기법을 처리하기 위해서 adaptation() 오퍼레이션을 제공하며, 제어 객체를 초기화하기 위해 initial() 오퍼레이션을 제공한다.

5.2.3 QoS 관리 모듈의 IDL 정의

QoS 관리 모듈을 구성하고 있는 제어 인터페이스 객체, 서비스 관리 객체, 접속 객체의 IDL 정의에 대해서 설명한다.

■ 제어 인터페이스 객체

다음의 (그림 30)은 제어 인터페이스 객체의 IDL을 나타낸다.

```
interface ControlInterface {
 attribute Connect conManager;
 attribute ServiceManagement smManager;
 boolean start();
 boolean stop();
 boolean setup(in qosparam parm);
 boolean getstatus(); };
```

그림 30. 제어 인터페이스 객체의 IDL

제어 인터페이스 객체는 QoS 관리 모듈을 구성하는 접속 객체와 서비스 관리 객체에 대한 객체 참조자를 관리하기 위해 conManager, smManager의 애트리뷰트를 유지하며, 접속 객체와 서비스 관리 객체의 활성화와 비 활성화를 위해 start() 오퍼레이션과 stop() 오퍼레이션을 제공하고, QoS 제어 관리 모듈의 제어 객체에서 상태 정보를 획득하기 위해 getstatus() 오퍼레이션을 제공한다. 그리고, 서비스 관리 객체에게 정보를 저장하기 위해 setup() 오퍼레이션을 제공한다.

■ 서비스 관리 객체

다음의 (그림 31)은 서비스 관리 객체의 IDL을 나타낸다.

```
interface ServiceManagement {
 boolean start();
 boolean stop();
 boolean setup(in qosparam param);
 qosparam getstatus();
 boolean initial(in UserInterface ui, in string remote);};
```

그림 31. 서비스 관리 객체의 IDL

서비스 관리 객체는 객체의 활성화와 비 활성화를 위해 start() 오퍼레이션과 stop() 오퍼레이션을 제공하고, QoS 파라미터를 저장하기 위한 setup() 오퍼레이션을 제공하며, 스트림 객체의 서비스 상태를 얻어오기 위한 getstatus() 오퍼레이션을 제공한다. 그리고 초기화를 위한 initial() 오퍼레이션을 제공한다.

■ 접속 객체

다음의 (그림 32)는 접속 객체의 IDL이다.

```
interface Connect {
  boolean start(in member receive, in member send);
  boolean stop();
  boolean join();
  boolean disjoin();
  boolean initial(in UserInterface ui, in string remote);};
```

그림 32. 접속 객체의 IDL

접속 객체는 스트림 송신 측 시스템에 접속하기 위해서 사용되는 start() 오퍼레이션을 제공하고, 서비스 세션을 종료와 탈퇴하기 위한 stop() 오퍼레이션과 disjoin() 오퍼레이션을 제공하며, 접속 객체의 전역 변수를 초기화하기 위한 initial() 오퍼레이션을 제공한다.

5.3 스트림 객체 구현

스트림 객체는 비디오 스트림의 전송만을 하는 것으로 제한하였으며, 기본 이미지 엔코더 형식은 JPEG 형식을 사용하였다. TCP를 이용해 RTP 패킷으로 전송하며, 해상도는 160×120, 320×240의 두 가지 크기를 지원하고, 스트림 전송 속도는 최대 30 프레임/초의 속도로 송수신 할 수 있도록 하였다.

스트림 서비스를 제어하기 위해 클라이언트 측의 사용자 인터페이스 객체를 통해 스트림 객체를 제어하도록 하며, 앞의 사용자 인터페이스의 클래스 다이어그램에서 나타낸 것과 같이 스트림 객체 인터페이스 중에서 start() 오퍼레이션은 스트림 객체가 스트림 데이터를 보내거나 받을 수 있도록 객체를 초기화 및 구동시키고, stop() 오퍼레이션은 스트림 객체를 정지시키고 할당된 자원을 해제하는 기능을 담당한다. setup() 오퍼레이션은 스트림 객체의 서비스 상태를 조절할 수 있는 인터페이스이고, getstatus() 오퍼레이션은 모니터링을 위해 스트림 객체로부터 스트림 파라미터(프레임 율, 해상도, CPU 사용량)를 얻어오는 기능을 담당한다.

5.4 분산 QoS 관리 구조의 실행 환경

본 절에서는 클라이언트 측에서 사용될 그래픽 사용자 인터페이스(GUI) 화면의 실행 환경과 실행 과정을 중심으로 기술한다. 먼저 그래픽 사용자 인터페이스 화면에 대해서 설명하고, 다음으로 클라이언트 측의 GUI 실행 환경을 통해 분산 QoS 관리 구조의 수행 과정과 제안된 QoS 제어 기법인 실시간 협약 및 동적 적응 과정을 나타낸다.

5.4.1 그래픽 사용자 인터페이스(GUI)

클라이언트 측의 그래픽 사용자 인터페이스 구현에 관한 내용으로 구현된 GUI를 중심으로 각각의 패널에 대해서 나타낸다. 다음의 (그림 33)는 스윙(Swing) 컴포넌트를 이용해 설계한 그래픽 사용자 인터페이스를 나타낸다.

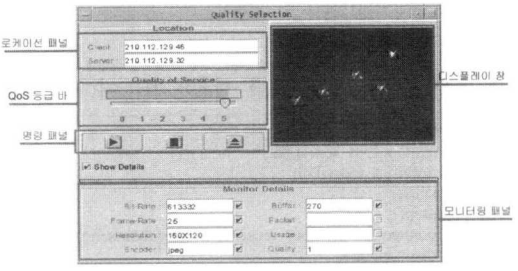


그림 33. 그래픽 사용자 인터페이스(GUI)

클라이언트 측에서 사용될 GUI는 서비스 접속에 사용하기 위한 각 모듈 객체의 호스트 주소를 지정할 수 있는 두 개의 텍스트 필드로 구성된 로케이션 패널(Panel)과 사용자에게 현재 QoS 등급을 표시하는 등급 바(Bar)가 있고, 스트림 서비스의 시작, 종료, 재설정 버튼으로 구성된 명령 패널이 있다. 그리고, 서버 측에서 송신되는 스트림 데이터를 클라이언트 측에서 수신하여 화면에 나타내는 디스플레이 창과 현재 비디오 스트림 수신 상태를 표시하는 모니터링 패널로 구성되며, 모니터링 패널은 GUI에 있는 Show Details라는 체크 박스를 누름으로써 나타나게 된다.

5.4.2 GUI 실행을 통한 제안된 QoS 제어 기법 검증

먼저 클라이언트 측의 사용자가 GUI 화면에 있는 시작 버튼을 누름으로써 서버 측으로부터 스트림 서비스를 요청한다. 요청을 받은 서버 측은 클라이언트의 요청에 따라 비디오 스트림 데이터를 전송함으로써 초기의 스트림 서비스가 시작된다. (그림 34)는 QoS 등급의 변화됨을 나타내고 있다.

초기 클라이언트 측과 서버 측의 협약된 QoS로 클라이언트에 비디오 스트림 데이터가 전송됨을 나

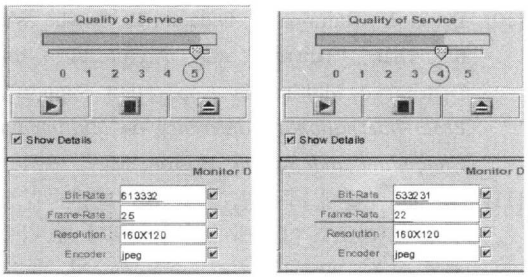


그림 34. 클라이언트 측의 QoS 등급 및 모니터링 데이터

타내고 있다. 이 후, 스트림 서비스 수행도중 클라이언트의 QoS 등급이 하부 네트워크 사정에 의해 한 단계 내려감을 나타내고 있다.

이와 같이 종단간 협약된 서비스 등급으로 서비스를 받지 못할 경우에 클라이언트의 QoS 등급이 내려감을 QoS 제어 관리 모듈에 있는 통보 객체가 이를 검출하여 제어 객체에게 알린다. 제어 객체는 매퍼 객체가 관리하는 DB에서 현재 상황에 맞는 QoS 파라미터 값을 QoS 관리 모듈의 제어 인터페이스 객체에게 넘긴다. 제어 인터페이스 객체는 서버 측의 데이터베이스에 모니터링 된 정보와 클라이언트 측에서 넘어온 정보를 비교하여 클라이언트의 현재 상황에 맞는 QoS 등급으로 송신중인 비디오 스트림 데이터를 재 설정한 후 클라이언트 측으로 스트림 데이터를 다시 송신한다. 다음의 (그림 35)는 클라이언트와 서버간에 실시간 협약 및 동적 적응 과정을 검증하기 위해 두 시스템의 DB 로직에 저장된 모니터링 된 정보를 바탕으로 QoS 등급 매핑 과정을 보이고 있다.

Ⅵ. 결론 및 향후 연구

인터넷의 IP 기술은 지역적인 한계를 넘어서 끝없이 많은 시스템과 통신 매체들을 연결하는 광역 네트워크의 형성을 가능하게 만들었다. 이는 전세계적으로 전자 우편과 웹 항해를 통해 연구, 학업, 업무 등을 수행하는 일상생활의 수단처럼 되었다. 또한 방송망과 전화망이 편리성, 보편성, 융통성을 좀 더 확장하기 위해서 IP와 통합을 시도하고 있으며, 이러한 새로운 네트워크의 출현들이 새로운 응용과 새로운 사용자들을 창출하는 견인차 역할을 하고 있다. 최근에는 이러한 다양한 멀티미디어 서비스를 이용하는 사용자의 기대가 더욱 커짐에 따라 이를 효율적으로 처리하고 제공하기 위하여 네트워크 상에서 분산 멀티미디어 응용간에 서비스 품질(QoS)은 매우 중요한 부분으로 인식되고 있다.

따라서, 본 논문에서는 분산 멀티미디어 환경인 CORBA 기반에서 자원들의 효과적인 관리와 다양한 서비스간의 QoS를 보장하는 분산 QoS 관리 시스템을 제시하였다. 이를 위해, QoS의 제어와 관리를 분산된 형태의 시스템으로 설계하였으며, 클라이언트 측에 QoS 제어 관리 모듈을, 서버 측에는 QoS 관리 모듈을 객체 형태로 구성하였다. 이는 분산 QoS 관리를 구조화함으로써 서버 시스템의 부하를 줄이고, 다양한 서비스 응용에 확장할 수 있도록 하며, 모듈의 재사용이 가능하도록 하였다. 마지막으로 분산 QoS 관리를 위한 제어 구조에서 QoS 제어기법은 실시간 협약과 동적 적응에 중점을 두어 보다 효과적인 QoS 제어기능을 제시했다.

향후 연구로는 세분화된 QoS 제어와 관리 기능을 위해서 다양한 알고리즘을 본 논문에서 제안하고 있는 분산 QoS 관리 구조에 적용시켜 보고, 좀 더 경량화된 분산처리 환경에 분산 QoS 관리 구조

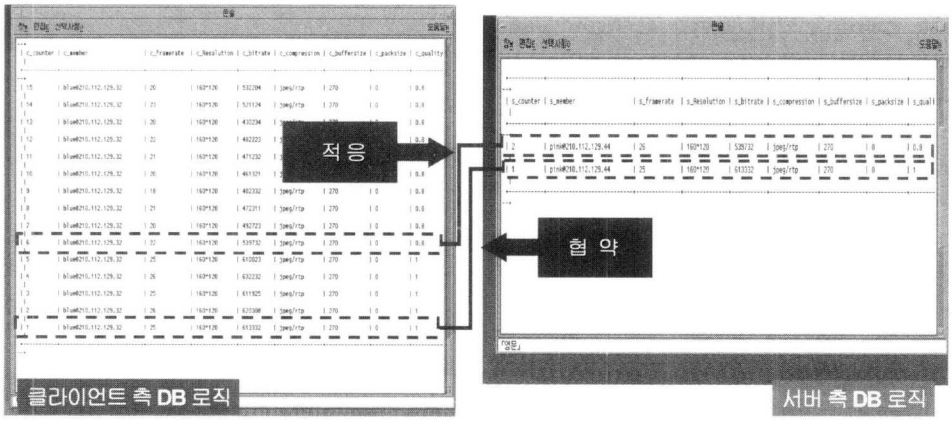


그림 35. 실시간 협약 및 동적 적응을 위한 두 시스템간의 매핑 테이블

를 적용하여 오디오/비디오 스트리밍 기술을 응용한 새로운 형태의 분산 멀티미디어 서비스 개발을 연구하고자한다. 또한, 분산 QoS 관리구조가 CORBA 기반의 분산 멀티미디어 응용에서 보다 효율적인 기능과 사용자 인터페이스를 갖도록 구현하고, 실제 인터넷 환경에서 QoS 만족도를 성능평가 하고자 한다.

참 고 문 헌

[1] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP:A transport protocol for realtime applications", *RFC-1889*. Feb, 1996

[2] H. Schulzrinne, "RTP Profile for Audio and Video Conference with Minimal Control", *RFC-1890*. May, 1996

[3] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the internet Architecture : An Overview" *RFC-1633*, June, 1994

[4] Hermann de Meer, Antonio Puliafito, Jan-Peter Richter, Orazio Tomarchio, "QoS-Adaptation by Software Agents in the Presence of Defective Reservation Mechanism in the Internet", Dept. of Computer Science University of Hamburg

[5] A. Lakas, G. S. Blair, A. Chetwynd, "A Formal Approach to the Design of QoS Parameters in Multimedia", Lancaster University Computing Department

[6] Carsten Vogt, Lars C. Wolf, Ralf Guido Herrtwich, Hartmut Wittig, "HeiRAT: Quality-of-Service management for distributed multimedia systems", *IBM European Networking Center*, 1998

[7] Nahrstedt K. and J. Smith, "Design, Implementation and Experiences of the OMEGA End-Point Architecture", Technical Report, University of Pennsylvania, May, 1995

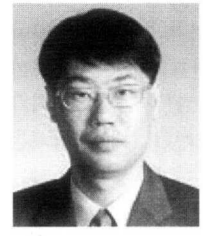
[8] ISO, "Quality of Service Framework", *ISO/IEC JTC1/SC21/WG1 N9680, International Standards Organization*, UK, 1995

[9] Ferrari, D., "The Tenet Experience and the Design on Protocols for Integrated Services Internetworks", *Multimedia System Journal*, 1996

[10] Besse, L., Dairaine L., Fedaoui, L., Tawbi, W., K. Thai, "Towards and Architecture for Distributed Multimedia Application Support", *Proc. International Conference on Multimedia Computing and System*, Boston, May, 1994

[11] 조동훈, 전병택, 주수종, "CORBA 환경의 분산 QoS 관리구조 기반에서 실시간 협약 및 적응 제어 연구", *한국정보처리과학회 학술지*, 제 27권 2호, 2000. 10.13 pp 8 - 14.

이 원 중(Won-Jung Lee) 정회원



1991년 2월 : 전북대학교 전산.통계학과(학사)
1998년 2월 : 원광대학교 대학원 컴퓨터공학과(석사)
1998년~현재 : 원광대학교 컴퓨터공학과 박사과정
2001년~현재 : 군장대학 전자상거래창업과 교수
<주관심 분야> 분산실시간컴퓨팅, QoS

신 창 선(Chang-Sun Shin) 정회원

1996년 2월 : 우석대학교 전산학과(학사)
1999년 8월 : 한양대학교 대학원 컴퓨터교육과(석사)
2000년~현재 : 원광대학교 컴퓨터공학과 박사과정
<주관심 분야> 실시간컴퓨팅, 분산객체모델

정 창 원(Chang-Won Jeong) 정회원

1993년 2월 : 원광대학교 컴퓨터공학과(학사)
1988년 2월 : 원광대학교 대학원 컴퓨터공학과(석사)
1998년~현재 : 원광대학교 컴퓨터공학과 박사과정
<주관심 분야>분산객체컴퓨팅, 멀티미디어데이터베이스

주 수 종(Su-Chong Joo) 정회원

1986년 2월 : 원광대학교 전자계산공학과(공학사)
1988년 2월 : 중앙대학교 대학원 컴퓨터공학과(공학석사)
1992년 2월 : 중앙대학교 대학원 컴퓨터공학과(공학박사)

1993년~1994년 : 미국 Univ. of Massachusetts at Amherst, 전기 및 컴퓨터공학과, Post-Doc.
1990년~현재 : 원광대학교 컴퓨터공학과 교수
<주관심 분야> 분산실시간컴퓨팅, 분산객체모델, 시스템최적화, 멀티미디어 데이터베이스