CORBA 기반 멀티미디어 스트림 통신 서비스의 설계 및 구현

김종현*, 정기동**
*동의공업대학 컴퓨터정보계열
**부산대학교 전자계산학과
e-mail : *jhkim@dit.ac.kr, **kdchung@melon.cs.pusan.ac.kr

Design and Implementation of CORBA based Multimedia Stream Communication Service

Jong-Hyun Kim*, Ki-Dong Chung**
*Dept. of Computer and Information, Dongeui Institute of Technology
**Dept. of Computer Science, Pusan National University

요 약

본 논문에서는 CORBA 환경 하에서 멀티미디어 스트림을 효율적으로 처리하고 제어하기 위한 멀티미디어 스트림 통신 서비스의 설계와 구현에 관한 내용을 기술한다. 제안하는 멀티미디어 스트림 통신 서비스에서는 멀티미디어 스트림의 효율적인 처리와 제어 그리고 전송을 위한 소프트웨어 구성 요소들을 분산 객체들로 설계한다. 그리고 제어 데이터와 미디어 데이터의 전송 경로를 분리하여 미디어 데이터의 전송은 RTP(Realtime Transport Protocol)로 직접적으로 전송하므로서 멀티미디어 스트림의 전송 효율을 최적화한다. 우리는 멀티미디어 스트림 통신을 위한 분산 객체들 간의 연결 설정과 제어 절차를 보여주며, 테스트 시스템을 구축하여 성능을 평가한다. 성능 평가 결과 연결 설정 지연은 TCP 연결에 비해 다소 지연을 가지나, 미디어 데이터의 전송은 CORBA 의 IIOP 프로토콜에 비해 최적화된 성능을 보여준다..

1. 서론

네트워크를 기반으로 하는 분산 시스템 환경에서 클라이언트, 서버 구조의 응용 프로그램을 구현할 때 소프트웨어나 하드웨어 플랫폼 등의 이질성으로 야기되는 상호운용성(interoperability) 문제는 응용 시스템의 개발을 매우 복잡하고 어렵게 한다. OMG(Object Management Group)에서 제안한 분산 객체 미들웨어인 CORBA(Common Object Request Broker)는 상호운용성 문제를 해결하며, 객체지향적인 개발 방법을 제공하여 표준 개방 분산 환경의 플랫폼으로 최근 주목을 받고있다[1].

CORBA 의 여러 가지의 장점에도 불구하고 현재 구현된 대부분의 CORBA 는 성능의 최적화 문제, 실시간 및 QoS(Quality of Service) 기능과 연속 미디어의 처리에 대한 기능 등의 부족으로 분산 멀티미디어 응용에 효율적으로 적용되기 어렵다[2]. OMG 에서도 이러한 점을 인식하고 최근 CORBA 상에서 멀티미디어 스트림을 효율적으로 처리하기 위한 RFP(Request For Proposal)을 제시하고 있다[3].

본 연구에서는 CORBA 환경 하에서 효율적으로 멀티미디어 스트림을 처리하고 제어하기 위한 소프트웨어 기반 구조인 멀티미디어 스트림 통신 서비스를 설계하고 구현한다. 멀티미디어 스트림 통신 서비스에서는 멀티미디어 스트림의 처리와 제어 그리고 전송을 지원하기 위한 분산 객체들로 설계한다. 그리고 멀티미디어 스트림의 전송 효율을 최적화하기 위하여 제어 데이터와 스트림 데이터의 전송 경로를 분리하여 미디어 데이터는 ORB(Object Request Broker)를 거치지 않고 RTP(Real Time Transport Protocol)를 통하여 직접적으로 전송한다. 설계한 멀티미디어 스트림 통신 서비스는 상용 CORBA 제품의 하나인 IONA 사의 Orbix 2000 과 SUN 사의 JMF(Java Media Framework) 2.0 사용하여 테스트 시스템을 구축하고 성능을 평가한다.

본 논문의 구성은 다음과 같다. 2 장에서는 본 논문과 관련된 연구 사례들을 살펴본다. 3 장에서는 제안하는 멀티미디어 스트림 통신 서비스의 설계 및 세부 사항에 대하여 기

술한다. 4 장에서는 3 장에서 언급한 설계를 바탕으로 테스트 시스템을 구현하고 시험한 내용에 대하여 설명한다. 마지막으로 5 장에서는 결론 과 향후 과제에 대하여 기술한다.

2. 관련 연구

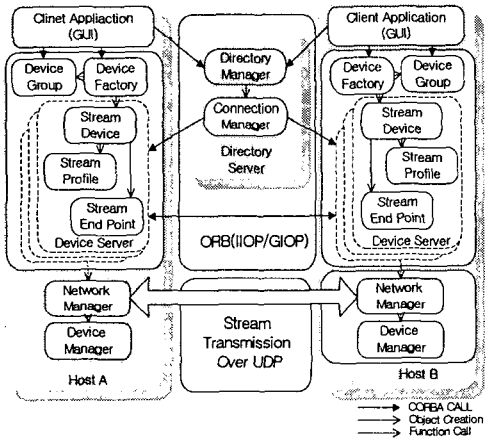
SUN 사의 JMF 나 Microsoft 사의 Netshow 등은 네트워크 환경에서 멀티미디어 스트림 데이터를 효율적으로 처리하기 위한 멀티미디어 응용 프로그램을 체계적으로 개발할 수 있도록 지원하고 있다[4]. 특히 JMF 는 플랫트폼의 독립성을 제공하고 객체지향적인 JAVA 프로그래밍 언어에서 화상회의나 VOD 등과 같은 멀티미디어 응용을 손쉽고 체계적으로 개발하기 위한 잘 정의된 API 를 제공한다. 그러나 JMF 나 Netshow 등은 투명한 분산 처리환경을 제공하는 소프트웨어 구조가 아니다.

OMG 에서 RFP 로 제안한 A/V(오디오/비디오) 스트림 서비스는 오디오/비디오 스트림의 제어와 관리에 관련된 소프트웨어 구성 요소들을 CORBA 객체로 모델링하고 IDL 로 정의하였다. OMG A/V 스트림 서비스는 표준에 기반한 분산 멀티미디어 응용의 개발을 용이하도록 하며, 다양한 멀티미디어 디바이스 유형과 다양한 미디어 형식을 지원하도록 설계되었다. 그리고 제어 데이터와 스트림 데이터의 전송 프로토콜을 분리하여, 스트림 데이터는 IIOP/GIOP 프로토콜과 독립적인 다른 적합한 프로토콜로 전송되도록 한다. OMG 의 A/V 스트림 서비스는 다소 복잡한 API 를 정의하고 있으며, 현재 일부 대학에서 실험적인 프로토타입으로 구현한 수준에 있으며, 상용화 및 표준화 단계에는 이르지 못하고 있다[5].

3. 시스템의 설계

3.1 시스템의 구조

본 논문에서 제안하는 멀티미디어 스트림 통신 서비스는 (그림 1)과 같이 인터넷에서 하나의 디렉터리 서버(Directory Server)와 여러 개의 클라이언트들로 구성되어 있다. 디렉터리 서버와 각 클라이언트 호스트에 분산되어 존재하는 분산 객체들은 스트림의 연결 설정과 전송 제어를 담당하는 CORBA 객체로 멀티미디어 스트림 통신 서비스를 위한 분산 제어 환경을 제공한다.



(그림 1) 멀티미디어 스트림 통신 서비스의 구조

디렉터리 서버에는 연결하고자 하는 디바이스들의 위치 정보를 제공해 주는 분산 객체인 디렉토리 관리자(Directory Manager)와 디바이스들간의 연결 설정과 전송 제어를 위한 연결 관리자(Connection Manager)가 존재한다. 연결 관리자는 하나의 스트림 전송 세션을 의미하며, 디렉토리 관리자 객체에 의하여 생성되어 관리된다

각 클라이언트 호스트에는 멀티미디어 디바이스의 제어에 관련된 사항을 정의한 객체들의 그룹인 디바이스 서버(Device Server)가 존재한다. 디바이스 서버는 디바이스의 일반적인 제어에 관한 사항을 정의하고 있는 스트림 디바이스(Stream Device) 객체, 디바이스 간 지원하고자 하는 미디어 형식을 정의하고 있는 스트림 프로파일(Stream Profile) 객체 그리고 멀티미디어 데이터의 네트워크 상의 전송에 관련된 사항을 정의하고 있는 Stream End Point 객체를 포함하고 있다. 디바이스 그룹(Device Grouo) 객체는 여러 개의 멀티미디어 스트림 전송 세션을 그룹화하여 일괄적으로 제어하기 위한 사항을 정의하고 있다. 시스템 내의 모든 디바이스 서버는 각 클라이언트 호스트 내에 디바이스 팩토리(Device Factory) 객체에 의해 생성되고 관리된다.

클라이언트 응용은 사용자 인터페이스를 포함하는 제어 응용으로 CORBA 클라이언트 형태로 존재한다. 디바이스 서버를 비롯한 시스템 내의 분산 객체들은 CORBA 서버로 형태로 존재하여 ORB 가 요구하는 형태인 IDL 로 표현된다. IDL 로 정의한 디바이스 서버의 구현은 클라이언트 호스트 내의 RTP 전송을 담당하는 네트워크 관리자(Network Manager)와 미디어의 캡쳐와 재생 그리고 물리적인 디바이스의 관리를 담당하는 디바이스 관리자(Device Manager)를 이용한다. 따라서 클라이언트 응용은 분산 시스템 상의 위치에 상관없이 존재가 가능하지만, 디바이스 서버들은 기본적으로 실제적으로 미디어 데이터를 캡쳐하거나 재생하는 물리적인 디바이스가 있는 호스트에 존재해야 한다.

클라이언트 응용과 분산 객체들 간 혹은 분산 객체 자체들 간의 연결 설정 및 전송 제어를 비롯한 모든 제어 데이터의 교환은 ORB 를 통해 원격 메소드 호출(RPC)로 이루어진다. 그러나 실시간 특성을 가지는 멀티미디어 스트림 데이터는 네트워크 관리자(Network Manager)에서 제공하는 별도의 RTP/UDP 통신 프로토콜을 통하여 직접적으로 송수신된다. RTP/UDP 통신 프로토콜로 송수신된 멀티미디어 스트림 데이터는 디바이스 관리자를 통하여 물리적인 디바이스로 전달된다.

3.2 스트림 통신을 위한 분산 객체

1) 디렉터리 관리자

디렉토리 관리자는 디렉토리 서버 내에 위치하여 디바이스 간의 연결을 위한 위치 정보를 제공하며, 연결 설정을 위하여 연결 관리자를 생성하고 관리하는 역할을 한다. 연결을 원하는 모든 클라이언트 응용은 먼저 각 클라이언트 호스트 내에 디바이스 서버 객체를 생성하고, 디렉토리 관리자에 바인드하여 사용자 정보(user ID)와 디바이스의 유형 그리고 디바이스 서버 객체의 레퍼런스(IOR: Interoperable Object Reference)를 등록한다.

디렉토리 관리자에 등록된 디바이스 서버의 레퍼런스와 사용자 정보 그리고 디바이스의 유형은 연결 리스트로 구성되어, 연결을 원하는 클라이언트 응용의 요구에 의해 디바이스 서버 간의 연결 정보를 제공한다

2) 디바이스 서버

디바이스 서버는 스피커나 마이크 등과 같은 물리적인 멀티미디어 디바이스의 제어와 처리에 관련된 사항을 정의한 객체들의 그룹으로 디바이스 팩토리, 스트림 디바이스, 스트림 프로파일, Stream End Point 그리고 디바이스 그룹 객체로 구성된다.

디바이스 팩토리는 각 클라이언트 호스트에 하나씩 존재

하여 클라이언트 응용의 요청에 의해 새로운 디바이스 서버들과 하나의 디바이스 그룹을 생성하고 관리하는 역할을 하는 Factory 객체이다. 시스템 내의 모든 디바이스 서버들은 디바이스 팩토리를 통하여 생성된다.

스트림 디바이스는 시스템의 실행 초기에 디바이스 팩토리 객체에 의해 하나의 미디어 스트림 전송 당 하나씩 생성되는 객체로 일반적인 멀티미디어 디바이스에 공통적으로 적용될 수 있는 제어 인터페이스를 가지고 있으며, 디바이스의 상태와 유형 그리고 사용자 정보를 유지한다. 그리고 관련 객체인 스트림 프로파일 객체와 Stream EndPoint 객체를 생성하고 관리하는 역할을 한다.

스트림 프로파일은 디바이스 간의 지원하고자 하는 미디어의 형식을 정의하고 있다. 디바이스 간의 연결 설정 시에 스트림 프로파일은 송수신 디바이스 간 미디어 형식이 상호 부합되는지 확인하고, 만약 상호 간 미디어 형식이 부합되지 않으면 연결은 실패하게 된다.

Stream End Point는 미디어 데이터의 전송에 관련된 사항을 정의한 객체로 미디어 스트림의 실질적인 연결 종단 점(end point)을 의미한다. Stream End Point 는 연결 설정 시에 상호간 객체의 레퍼런스를 교환하고, 미디어 데이터 전송을 위한 UDP 네트워크 정보(IP 주소, 포트 번호)를 상호 교환하여 네트워크 프로토콜 간의 연결을 한 후, 미디어 데이터가 RTP/UDP 네트워크 프로토콜을 통하여 전송을 시작하거나, 중단하도록 한다.

디바이스 그룹은 다수개의 미디어 스트림 전송 세션을 그룹화하여 전체 세션을 일괄적으로 제어하기 위한 객체이다. 이를 위하여 특정 그룹에 속하는 각 스트림 디바이스들의 레퍼런스를 저장하기 위한 송수신 측의 참여 리스트를 유지하며, 스트림 디바이스가 새로운 멤버로 그룹에 참여(join)하거나, 기존의 그룹에서 탈퇴(leave)할 수 있도록 한다.

3) 연결관리자

연결 관리자는 클라이언트 응용에게 연결을 원하는 디바이스 서버간의 연결 설정과 전송 제어 인터페이스를 제공하는 객체로 하나의 스트림 전송 세션을 의미한다. 클라이언트 응용으로 부터의 연결 설정과 디바이스의 제어 그리고 전송 제어 요청은 연결관리자를 통하여 송수신 측의 디바이스 서버들에게 전달되어 디바이스 간의 연결(connect)과 연결 해제(disconnect) 그리고 디바이스의 시작(start)과 중지(stop) 등의 제어를 한다.

3.3 분산 객체를 통한 스트림 통신 절차

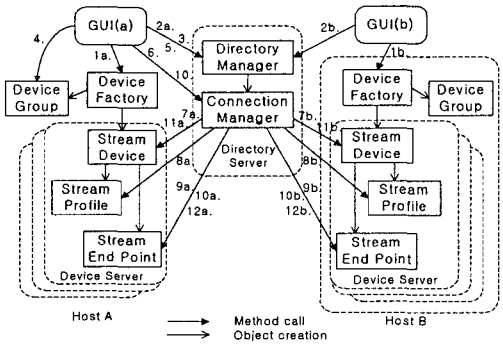
(그림 2)는 클라이언트 응용(GUI)과 분산 객체들 간의 연결과 제어 절차를 보여 준다. 송신측 호스트(Host A)에서 수신측 호스트(Host B)로 미디어 스트림을 전송하기 위한 스트림 제어 응용은 송신측 GUI(A)에 포함된다고 가정한다. 시스템의 초기 상태는 송수신측의 클라이언트 호스트 내에 CORBA 서버인 디바이스 팩토리와 디렉토리 서버 내의 디렉토리 관리자에서 클라이언트 응용으로부터 요청이 오기를 기다리고 있는 상태이다.

단계 1) GUI(a)와 GUI(b)는 자신의 지역 호스트(Host A, Host B) 내의 디바이스 팩토리에 바인드하여 연결 하고자 하는 스트림 디바이스들과 하나의 디바이스 그룹을 생성하고, 스트림 디바이스의 레퍼런스를 디렉토리 관리자에 등록한다 (1a.1b.2a.2b)

단계 2) 제어 응용인 GUI(a)는 디렉토리 관리자로부터 연결하고자 하는 송수신 측 스트림 디바이스 의 등록 여부를 확인하고 레퍼런스를 획득한다. 디렉토리 관리자로부터 획득한 송수신 측의 스트림 디바이스의 레퍼런스들은 다바이스 그룹에 참여(join)하여 디바이스 서버들 간의 그룹을 형성한다(3.4.).

단계 3) 디렉토리 관리자는 연결 설정을 위하여 연결관리자를 생성한다(5).

단계 4) GUI(a)는 디바이스 그룹 객체에 등록된 송수신 측의 스트림 디바이스의 레퍼런스 리스트를 인자로 연결관리자의 연결 (connect) 메소드를 호출하여 연결에 참여한 디바이스 들에 대하여 연결 설정을 시작한다(6).



(그림 11) 연결 설정 과정

단계 5) 연결관리자는 디바이스 서버 그룹에 참여한 스트림 디바이스들에게 StreamEndPoint 와 스트림 프로화일을 각각 생성하도록 한다.(7a, 7b).

단계 6) 연결관리자는 송수신 측의 스트림 프로화일 객체를 통하여 상호간 지원 가능한 미디어 형식을 확인한다. 이때 상호간 지원되는 미디어 형식이 부합되지 않을 경우 연결은 종료된다(8a, 8b).

단계 7) 연결관리자는 StreamEndPoint 를 통하여 상호간의 레퍼런스를 교환하고, 네트워크 프로토콜간의 연결을 위하여 네트워크 정보를 교환하여 연결 설정을 완료한다(9a, 9b).

단계 8) 연결 설정이 완료되면 GUI(a)는 연결 관리자를 통하여 디바이스를 준비시킨후, Stream End Point 를 통하여 실질적인 미디어 스트림의 전송을 시작하도록 한다.(10,11a,11b,12a,12b).

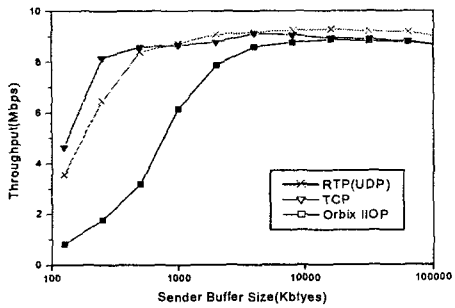
4. 구현 및 평가

테스트 시스템을 구현하기 위한 하드웨어 플랫폼으로 디렉터리 서버는 Solaris 7 을 운영체제로 사용하는 SUN Ultra 10 을 사용하였으며, 클라이언트 호스트는 화상 카메라 장치와 마이크 장치를 부착한 Windows NT 4.0 을 운영체제로 사용하는 Pentium PC 를 대상으로 하였다. 디렉터리 서버와 각 클라이언트 호스트에는 모두 IONA 사의 CORBA 소프트웨어인 Orbix 2000[6]이 탑재되어 있으며, Gigabit 이더넷 백본에 연결되어 있다. 디렉토리 관리자와 연결 관리자 는 Solaris 용 JAVA 로 구현하며, 클라이언트 호스트 내의 분산 객체들은 Windows 용 JAVA 로 구현된다. 멀티미디어 데이터의 캡쳐와 재생 그리고 전송 채널 등 디바이스에 의존적인 기능들은 SUN 사의 JMF 에서 제공하는 클래스들을 확장하여 구현한다. 테스트 시스템은 멀티미디어 스트림 통신 서비스에서 제공하는 분산 객체들에 정의한 메소드들을 이용하여 하나의 오디오 송 세션과 비디오 전송 세션으로 구성하였다.

테스트 시스템의 미디어 데이터의 전송 성능을 평가하기 위하여 RTP(UDP)와 TCP 프로토콜 그리고 Orbix 의 IIOP 프

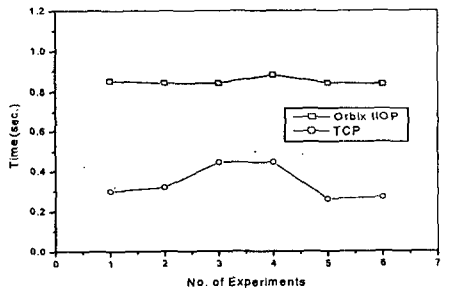
로토콜로 전송되는 데이터의 전송 처리율을 측정하였다. 동일한 환경에서 데이터를 전송할 때 송신측의 버퍼 크기 변화(128 byte ~ 128 Kbyte)에 따른 전송 처리율을 측정하였다. 이때 전송되는 데이터는 8 bit 의 octet 형식의 데이터 스트림을 사용한다. Orbix IIOP 프로토콜로 전송되는 octet 데이터 스트림은 oneway 호출을 통하여 수신측에게 전달된다. 송신측에서 10 분 동안 연속적으로 데이터를 전송할 때 수신측에서 전달 받은 octet 데이터의 합을 x 라고 하면 전송 처리율(T)는 식(1)과 같이 정의된다.

$$T = \frac{(x \times 8)}{(10 \times 60)} \times 10^{-6} (Mbps) \qquad (1)$$



(그림 3) 처리율 결과

(그림 3)은 RTP(UDP)와 TCP 그리고 Orbix IIOP 프로토콜을 사용한 전송 처리율 결과를 보여준다. RTP(UDP) 프로토콜을 사용한 데이터의 전송 성능이 TCP 보다 약간 우수함을 볼 수 있다. 그러나 Orbix IIOP 프로토콜의 경우 평균적으로 가장 낮은 전송 성능을 나타낸다. 이러한 이유는 Orbix 의 전송 계층인 ORB 는 과도한 동적 메모리 할당과 자료의 복사 그리고 marshalling/demarshaling 등과 같은 오버헤드로 인하여 전체 전송 성능을 현저히 저하시키기 때문이다[5]. 따라서 실시간 데이터인 미디어 데이터는 CORBA 의 ORB 전송 계층의 오버헤드를 피하기 위하여 독립적인 RTP/UDP 프로토콜로 전송하는 것이 바람직함을 알 수 있다.



(그림 4) 연결 설정의 지연

(그림 4)는 Orbix IIOP 와 TCP 프로토콜을 사용한 연결 설정 지연 시간을 보여준다. 연결 설정 지연 시간은 클라이언트 응용에서 디렉토리 서버로부터 연결하고자 하는 두개의 디바이스에 대한 객체의 레퍼런스를 획득한 순간부터 디바이스 간의 네트워크 전송 계층의 TCP 연결이 완료되는 시점까지의 시간을 측정하였다. Orbix IIOP 를 통한 연결 설정의 경우, TCP 를 통한 연결 설정의 경우보다 평균 0.4 sec. 지연이 발생되는 것을 알 수 있다. 그러나 제어 데이터인 연결 설정 데이터는 미디어 데이터 전송량에 비해 매우 작으며, 연결 후에는 미디어 데이터는 ORB 와는 독립적인 경로로 처리되어 RTP(UDP) 프로토콜을 통하여 미디어 데이터를 전송되므로 전체 시스템의 성능에는 크게 영향을 받지 않을 것으로 사료된다.

5. 결론 및 향후과제

본 연구에서는 분산 객체 미들웨어인 CORBA 를 기반으로 멀티미디어 스트림을 효율적으로 처리하고 제어하기 위한 멀티미디어 스트림 통신 서비스를 설계하고 구현하였다. 제안한 멀티미디어 통신 서비스에서는 멀티미디어 스트림의 효율적으로 제어와 처리 그리고 전송을 위한 분산 객체들을 새로이 정의하여 응용 개발자에게 직관적이고 간단한 메소드들을 제공하여 플랫폼과 프로그래밍 언어에 독립적인 분산 멀티미디어 응용 개발을 용이하게 한다. 그리고 제어 데이터와 미디어 데이터의 전송경로를 분리하여 실시간 데이터인 미디어 데이터의 전송은 RTP 프로토콜을 통하여 전송하고, 비실시간 데이터인 모든 제어 데이터는 ORB 를 통하여 전송하여 시스템의 전송 효율을 최적화한다. 설계한 멀티미디어 스트림 통신 서비스는 하나의 오디오 전송 세션과 비디오 전송 세션을 가지는 테스트 시스템을 구축하여 미디어 데이터의 전송 효율과 연결 설정 지연 시간 등의 실험하였다. 실험 결과 연결 설정 지연은 일반적인 TCP 연결에 비해 다소 지연을 가지나, 대용량이며, 실시간 데이터인 미디어 데이터의 전송은 RTP/UDP 프로토콜을 사용하므로 서 CORBA 의 ORB IIOP 프로토콜보다 월등한 전송 성능을 보여주었다.

제시한 시스템에서는 네트워크의 부하에 따른 미디어 데이터의 손실이나 재전송 그리고 미디어 코덱의 변경 등 QoS 관리 기능을 제공하지 못하고 있다. 따라서 RTP 에서 제공하는 제어 메시지인 RTCP 를 이용하여 네트워크과 시스템의 상태에 따른 동적인 QoS 서비스를 담당하는 분산 객체들에 대한 연구가 필요하다.

본 연구 결과를 토대로 향후 인터넷 상에서 화상회의, 원격 교육 등 다양한 분산 멀티미디어 응용에 적용하고자 한다.

참고문헌

[1] Jon Siegel, "CORBA Fundamental and Programming", John Wieley & Sons, Inc., 1996.

[2] Frank Siqueria, 1998, "The Design of a Generic QoS Architecture for Open Systems", Internal Report, Distributed Systems Group, Trinity College Dublin

[3] Object Management Group, 1997, "Control and Management of A/V Stream specification", OMG Document telecom/97-05-07, 1997

[4] Java Media Framework API, http://java.sun.com/products/java-media/jmf/, Sun Microsystems Inc. 1998

[5] Mungee S., Surendran N. and Schmidt D.C., "The Design and performance of a CORBA Audio/Video Stream Service", HICSS-32, Hawaii, Jan. 1999.

[6] IONA Technologies PLC, "Orbix Programmer's Guide", IONA Technologies Ldt., 1997.