분산 멀티미디어 응용 프로그램을 위한 통신 서비스

1998

윤태형

본 학위 논문 내용에 관하여 학술, 교육 목적으로 사용할 모든 권리를 포항공대에 위임함

석사학위논문

분산 멀티미디어 응용 프로그램을 위한 통신 서비스

윤태형 (尹泰亨)
전자계산학과
포항공과대학교
1998

분산 멀티미디어 응용 프로그램을 위한 통신 서비스

Communication Service for Distributed Multimedia Applications

Communication Service for Distributed Multimedia Applications

by

Tae-Hyoung Yun

Department of Computer Science & Engineering
POSTECH

A thesis submitted to the faculty of POSTECH in partial fulfillment of the requirements for the degree of Master of Science in the Department of Computer Science & Engineering

Pohang, Korea
November 27, 1997
Approved by

_____
Major Advisor

분산 멀티미디어 응용 프로그램을 위한 통신 서비스

윤태형

위 논문은 포항공과대학교 전자계산학과 석사학위논문으로 학위논문 심사위원회를 통과하였음을 인정함.

1997 년 11 월 27 일

학위논문심사 위원회 위원장 홍원기 (인)

위 원 강교철 (인)

위 원 김정현 (인)

MCOS
9625M22

윤태형, Tae-Hyoung Yun, Communication Service for Distributed Multimedia Applications, 분산 멀티미디어 응용 프로그램을 위한 통신 서비스, 전자계산학과, 1998, 42P, Advisor : Won-Ki Hong, Text in Korean.

ABSTRACT

Multimedia applications are being developed and used for many aspects of our lives today. New high-speed, broadband networks have emerged and made the operation of these high-bandwidth requiring applications readily feasible. However, the development of distributed multimedia applications and efficient and reliable operation of these applications are still very difficult. This is because most conventional operating systems do not provide multimedia-related services and current networking technologies do not satisfy the requirements of various multimedia applications (such as multi-point communication, real-time processing, media synchronization, etc.).

In order to solve this problem, a system called MAESTRO has been developed. MAESTRO provides a rich multimedia API and distributed multimedia services such as communication service, name service, session service, storage/retrieval service and management service that can be used to develop a variety of multimedia applications easily.

Among the distributed multimedia services, this thesis presents the design and implementation of the communication service that helps multimedia applications communicate each other efficiently and reliably. The communication service has been modeled using object-oriented methodology and consists of various distributed object definitions. The functionalities provided by the communication service include multi-point communication, flow control, reliable communication, bandwidth control.

As a proof of concept, this thesis presents how a prototype of the communication service has been implemented using a CORBA distributed platform and how the distributed multimedia applications such as video/audio conferencing tool and vide/audio multicasting tool can be developed using the communication service.

목차

그림 목차

1. 서론

멀티미디어는 앞으로 교육, 사업, 건강 정보, 출판, 게임 등 우리 생활에 직접적으로 영향을 미치며 우리 생활의 한 부분으로 중요한 역할을 차지할 것이다. 현재 멀티미디어와 관련된 광범위한 연구가 진행 중에 있고 머지 않아 다양한 멀티미디어 응용 프로그램들을 우리 생활 주변에서 접하게 될 것이다. 최근의 컴퓨팅 환경과 네트워킹 기술의 발전은 실시간 처리나 높은 대역폭을 요구하는 다양한 멀티미디어 응용 프로그램들의 출현을 더욱 가속화시키고 있다.

그런데 멀티미디어 응용 프로그램들이 효율적으로 사용되도록 하려면 지리적으로 멀리 떨어져 있는 사용자들이 멀티미디어 응용 프로그램들을 이용해서 서로 친밀하게 통신할 수 있어야 한다. 즉, 전세계 사용자들이 마치 한 곳에 있는 것처럼 서로 멀티미디어 응용 프로그램을 이용해서 통신할 수 있어야 한다. 그러나 현재의 컴퓨팅 환경과 네트워킹 기술은 이와 같은 멀티미디어 응용 프로그램의 조건을 만족시키기에 충분하지 못하다. 대부분의 운영 체제들은 멀티미디어 응용 프로그램들을 효율적으로 지원할 수 있는 서비스를 제공하지 못한다. 네트워킹 기술은 아직도 멀티미디어 응용 프로그램이 요구하는 화상 회의, 공동 작업, 멀티캐스팅 등에 적합한 통신 체계를 제공하지 못하고 있다.

현재의 컴퓨팅 환경과 네트워킹 기술이 멀티미디어 응용 프로그램들의 요구 사항을 제대로 충족시켜주지 못하기 때문에 멀티미디어 응용 프로그램들을 시스템 차원에서 지원해 주려는 연구들 [8, 9, 10, 11, 12]이 있었다. 그러나 이 연구들은 특정 멀티미디어 응용 프로그램을 목적으로 연구 중이거나 아직까지 연구가 진행 중인 경우가 많았다. 따라서 다양한 멀티미디어 응용 프로그램들을 개발하기 위해서 이 연구들을 직접 이용할 수가 없었다.

이런 이유에서 다양한 멀티미디어 응용 프로그램들을 지원할 수 있고 직접 이용할 수 있는 시스템에 대한 연구를 시작하였는데 이 시스템을 MAESTRO [1, 2, 3, 4]라 한다. MAESTRO 는 다양한 멀티미디어 응용 프로그램들을 지원하기 위해서 멀티미디어 응용 프로그램들을 쉽게 개발할 수 있도

록 하고 개발된 응용 프로그램들을 안정적이고 효율적인 방법으로 운용, 관리될 수 있도록 해 준다. MAESTRO 는 **멀티미디어** API (Application Programming Interface)층과 **분산 멀티미디어 서비스** 층으로 구성되는데 멀티미디어 API 층은 멀티미디어 응용 프로그램들이 공통적으로 이용할 수 있는 여러 클래스 정의들로 구성되며 분산 멀티미디어 서비스 층은 다양한 서비스를 제공해주는 **분산 객체**들로 구성된다. 여기서 분산 객체란 CORBA [13, 14, 15]나 DCOM [16]과 같은 분산 객체 기술을 이용하여 서비스를 제공해 줄 수 있는 객체를 의미한다. 분산 멀티미디어 서비스 층에서 제공해 주는 서비스들로 **통신 서비스** (Communication Service), **네임 서비스** (Name Service), **세션 서비스** (Session Service), **저장/인출 서비스** (Storage/Retrieval Service), **관리 서비스** (Management Service) 등이 있다.

이 논문은 MAESTRO 가 제공하는 분산 멀티미디어 서비스들 중 통신 서비스의 설계와 구현에 대해서 설명한다. MAESTRO 통신 서비스는 멀티미디어 응용 프로그램들이 서로 효율적이고 안정적으로 통신할 수 있도록 한다는 것을 목표로 하며 다자간 통신 (Multi-point Communication), 흐름 제어 (Flow Control), 신뢰성 있는 통신 (Reliable Communication), 대역폭 제어 (Bandwidth Control) 등의 기능을 제공해 준다.

이제 이 논문은 다음과 같은 순서로 설명된다. 먼저 2 장에서는 MAESTRO 통신 서비스와 관련 있는 기존의 연구들을 살펴본다. 3 장에서는 MAESTRO 통신 서비스에 대해서 설명하기 전에 MAESTRO 통신 서비스가 제공되는 시스템인 MAESTRO 에 대해서 개략적으로 설명한다. 4 장에서는 멀티미디어 응용 프로그램들의 MAESTRO 통신 서비스에 대한 요구 사항에 대해서 설명한다. 5 장에서는 이 논문의 핵심인 MAESTRO 통신 서비스에 대해서 설명하는데 통신 서비스를 구성하는 여러 분산 객체들에 대해서 설명하고 이 분산 객체들이 다자간 통신, 흐름 제어, 신뢰성 있는 통신, 대역폭 제어 등의 기능을 제공해 주기 위해서 서로 어떻게 동작하는가에 대해서 설명한다. 6 장에서는 MAESTRO 통신 서비스의 구현에 대해서 설명하고 7 장에서는 MAESTRO 통신 서비스를 이용해서 구현된 멀티미디어 응용 프로그램으로 화상 회의 도구와

비디오/오디오 멀티캐스팅 도구에 대해서 설명한다. 마지막으로 8 장에서는 결론을 맺고 향후 계획에 대해서 설명한다.

2. 관련 연구

일반적으로 통신 서비스가 제공해 주어야 하는 중요한 기능들로 다자간 통신 (Multi-point Communication), 흐름 제어 (Flow Control), 신뢰성 있는 통신 (Reliable Communication), 대역폭 제어 (Bandwidth Control), 미디어 동기화, 실시간 통신 등이 있다.

다자간 통신은 멀티캐스팅 (multicasting)이라고 할 수도 있으며 화상 회의 도구와 같은 공동 작업을 위한 멀티미디어 응용 프로그램의 개발에 중요한 기능이다. 다자간 통신은 일대일 통신 (One-to-One Communication), 일대다 통신 (One-to-Many Communication), 다대일 통신 (Many-to-One Communication), 다대다 통신 (Many-to-Many Communication)의 기능들을 제공해야 한다. 흐름 제어는 송신 프로그램이 데이터를 전송한 순서대로 수신 프로그램이 데이터를 받을 수 있도록 해 주는 기능이다. 신뢰성 있는 통신은 송신 프로그램이 보낸 데이터가 전송 도중에 사라지지 않고 수신 프로그램에게 전달되도록 해 주는 기능이다. 패킷 전송 네트워크일 경우 네트워크상의 부하 (load)가 심하면 전송되는 패킷이 소실될 수 있는데 이런 경우 패킷의 재전송 등을 통해서 신뢰성 있는 통신을 보장해 주어야 한다.

대역폭 제어는 멀티미디어 응용 프로그램들이 자신이 원하는 통신 대역폭을 요청할 수 있고 이 통신 대역폭이 멀티미디어 응용 프로그램의 통신 시간 동안 보장될 수 있도록 해 주는 기능이다. 미디어 동기화는 여러 미디어들을 여러 채널들을 이용해서 동시에 전송할 경우 미디어들을 시간에 따라 동기화 시켜주는 기능이다. 실시간 통신은 송신 프로그램이 보낸 데이터가 특정 시간이 지나기 전에 수신 프로그램에게 전달되도록 해 주는 기능이다.

이제 이 절에서는 위에서 설명한 여러 기능들이 StreamWorks [36]와 RealAudio [37], IP 멀티캐스트 [17]와 MBONE [20] 등에서 어떻게 고려되고 있는가를 순서대로 설명한다.

2.1. StreamWorks 와 RealAudio

StreamWorks [36]와 RealAudio [37]는 **응용 프로그램 수준**에서 멀티캐스

팅 기능을 제공하고 있는데 이것은 송신 프로그램이 각 수신 프로그램에게 일일이 데이터를 전달해 주는 방법이다. 이 방법은 전송측 컴퓨터의 정보 처리 양이 많고 네트워크를 통해서 각 수신 프로그램에게 동일한 데이터가 중복적으로 전달되기 때문에 네트워크 대역폭의 낭비가 심해서 많은 수신 프로그램을 포용할 수 없다. 흐름 제어, 신뢰성 있는 통신, 대역폭 제어, 미디어 동기화, 실시간 통신 등의 기능들도 응용 프로그램 수준에서 제공되거나 고려되지 않은 경우가 많다.

이 방법은 응용 프로그램 수준에서 모든 기능들을 제공하기 때문에 여러 멀티미디어 응용 프로그램들이 이 기능들을 공통으로 사용할 수 없다.

2.2. IP 멀티캐스트와 MBONE

StreamWorks 와 RealAudio 와 같이 응용 프로그램 수준에서 멀티캐스팅 기능을 제공하는 방법들은 데이터를 효율적으로 전송할 수 없다. 따라서 이런 단점을 제거하기 위한 대안으로 **네트워크 수준**에서 멀티캐스팅 기능을 제공하여 데이터를 효율적으로 전송하려는 연구로 **IP 멀티캐스트** [17]와 MBONE [20]이 있다.

IP 멀티캐스트는 인터넷의 네트워크 층에서 사용되는 기술로 송신자는 수신자의 수에 상관 없이 데이터를 전송하면 네트워크 자체가 필요에 의해서 데이터를 복제하고 각 수신자에게 데이터를 전달해 주는 기술이다. IP 멀티캐스트에서는 네트워크 스스로 데이터를 복제하고 전달하도록 하기 위해서 네트워크 자체가 송신자를 루트 (root)로 하고 수신자들을 터미널 (terminal)로 하는 **멀티캐스트 트리** (multicast tree)를 만들어서 관리한다.

IP 멀티캐스트는 Distance Vector Multicast Routing Protocol (DVMRP) [18]과 Internet Group Management Protocol (IGMP) [19]의 두 가지 프로토콜로 구성된다. DVMRP 는 IP 멀티캐스트를 지원하는 라우터들 사이에서 사용되는 프로토콜로 송신자를 루트로 하는 멀티캐스트 트리를 생성하고 관리하기 위해서 사용된다. 라우터들은 하나의 송신자에 하나의 멀티캐스트 트리를 생성해서 관리하는데 송신자가 패킷을 보내면 라우터는 그 송신자에 대한 멀티캐스트 트리 정보를 이용해서 패킷을 라우트하여 패킷이 각 수신자에게 전달되도록 한다. IGMP 는 수신자와 수신자로부터 가장 가까운 라우터

사이에 이용되는 프로토콜로 수신자가 **멀티캐스트 그룹** (multicast group)에 참여하거나 참여하고 있는 멀티캐스트 그룹을 떠나고자 할 때 이용된다. IGMP 를 이용해서 수신자가 멀티캐스트 그룹에 참여하거나 떠나면 멀티캐스트 트리에 변화가 생긴다.

수신자가 특정 멀티캐스트 그룹을 지정할 수 있도록 하기 위해서 IP 어드레스 중의 일부를 예약하여 사용하는데 이것을 **멀티캐스트 그룹 어드레스** (multicast group address)라고 한다. 송신자는 특정 멀티캐스트 그룹 어드레스를 이용해서 데이터를 보내고 수신자는 자신이 원하는 멀티캐스트 그룹 어드레스를 선택해서 해당 멀티캐스트 그룹에 참여함으로써 데이터를 받을 수 있다. 인터넷 IP 어드레스 중에서 멀티캐스트 그룹 어드레스로 예약되어 있는 어드레스들은 D 클래스 IP 어드레스들이다 [19].

그림 1 은 라우터들과 송신자와 수신자들을 포함하는 멀티캐스트 트리의 예를 보여 준다. 그림 1 에서 송신자가 패킷을 보내면 라우터 A 가 그 패킷을 받고 라우터 A 는 송신자를 루트로하는 멀티캐스트 트리를 참조하여 라우터 B, C 와 수신자 A 에게 패킷을 전달한다. 라우터 B 와 C 도 각각 멀티캐스트 트리를 참조하여 해당 수신자 B, C 와 D 에게 패킷을 전달한다. 이처럼 멀티캐스트 트리에서 패킷이 전달될 때 단지 한 카피 (copy)만 전달되므로 네트워크 자원을 낭비하지 않는다.

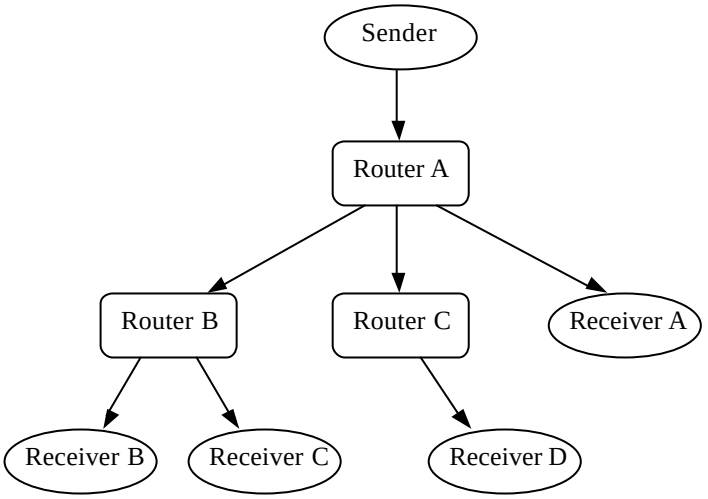


그림 1. 멀티캐스트 트리의 예

IP 멀티캐스트에서 송신자는 어떤 수신자들이 데이터를 받고 있는 지를 알 수가 없다. 송신자는 단지 특정 멀티캐스트 그룹 어드레스를 이용해서 데이터를 전송할 뿐이며 수신자가 해당 멀티캐스트 그룹에 참여하거나 떠나더라도 송신자에게 그 사실이 알려지지 않는다. 이것은 IP 멀티캐스트를 이용하여 송신자가 데이터를 전송할 경우 송신자는 수신자들에 대해서 어떤 제한을 가할 수 없으며 불특정 다수의 수신자들이 그 데이터를 받아 보게 된다는 것을 의미한다. 따라서 IP 멀티캐스트는 특정 목적을 가지는 소그룹의 사용자들을 위한 응용 프로그램의 개발에 적합하지 않고 전세계 불특정 다수를 위한 응용 프로그램의 개발에 적합하다.

1994 년에 인터넷 상에 실제로 존재하는 몇몇 라우터들이 IP 멀티캐스트를 지원할 수 있도록 만들어졌고 이 라우터들은 IP 터널을 이용해서 연결되면서 유니캐스팅만을 지원할 수 있는 물리적인 네트워크 위에서 멀티캐스팅을 지원할 수 있는 가상적인 네트워크가 실현되기 시작했다. 이 멀티캐스팅을 지원할 수 있는 가상적인 네트워크를 멀티캐스트 백본 (Multicast Backbone) [20] 또는 MBONE 이라 부르며 현재 전 세계적으로 널리 퍼져 있는 상태이다.

그러나 MBONE 은 흐름 제어, 신뢰성 있는 통신, 대역폭 제어, 미디어 동기화, 실시간 통신 등의 기능들을 제공하지 못한다. MBONE 에서는 IP 패킷을 멀티캐스트하기 때문에 보통 User Datagram Protocol (UDP) [21] 프로토콜이 이용된다. 따라서 네트워크 자체가 흐름 제어를 해 주지 못하며 수신 프로그램이 직접 흐름 제어를 해야 한다. 또, 신뢰성 있는 통신을 보장하지도 못하기 때문에 수신자는 불완전한 데이터를 받을 수도 있다. 따라서 수신자가 불완전한 데이터를 의미 있는 데이터로 복구할 수 있도록 하려면 송신자와 수신자가 사전에 약속한 어떤 방법이 필요하다.

2.3. 신뢰성 있는 멀티캐스팅 프로토콜들

IP 멀티캐스트가 신뢰성 있는 통신을 제공해 주지 못하기 때문에 IP 멀티캐스트 위에서 신뢰성 있는 통신을 보장해 주기 위한 프로토콜에 대한 연구들이 있었는데 Reliable Multicast Protocol (RMP) [22], Scalable Reliable Multicast (SRM) [23], Log-Based Reliable Multicast (LBRM)

[24], Reliable Multicast Transport Protocol (RMTP) [25] 등의 프로토콜이 이에 해당한다.

2.4. 실시간 전송 프로토콜

Real-time Transport Protocol (RTP) [26]은 IETF 에 의해서 표준으로 정해진 프로토콜로 멀티캐스트 또는 유니캐스트를 이용해서 실시간 데이터를 효율적으로 보낼 수 있도록 해 주는 실시간 전송 프로토콜이다.

RTP 를 이용하는데 있어 다양한 제어 기능을 제공하기 위하여 별도의 프로토콜로 RTP Control Protocol (RTCP)을 이용한다. RTP 가 데이터 전송에만 관계하는데 비해 RTCP 는 데이터 전송을 모니터하고 세션에 관련된 정보를 주고 받는데 이용된다. RTP 를 이용한 데이터 전송은 RTCP 를 이용한 제어 정보에 의해서 영향을 받는다.

RTP 는 현재 MBONE 에서 이용되고 있으며 이를 이용한 응용 프로그램들이 점차 많아 지고 있다.

3. MAESTRO 개요

이 장에서는 통신 서비스가 운용되는 시스템인 MAESTRO 에 대해서 개략적으로 설명한다.

3.1. MAESTRO 의 구조

그림 2 는 MAESTRO 의 계층적 구조를 보여 준다. 맨 위 층에 화상 회의, Video On Demand (VOD), 텔레메디슨, 화이트 보드 등 다양한 멀티미디어 응용 프로그램들이 존재한다. 멀티미디어 응용 프로그램들은 MAESTRO 상에서 여러 구성 객체들로 분산될 수 있으며 분산된 각 객체들이 서로 상호 작용하면서 하나의 응용 프로그램을 구성할 수 있다.

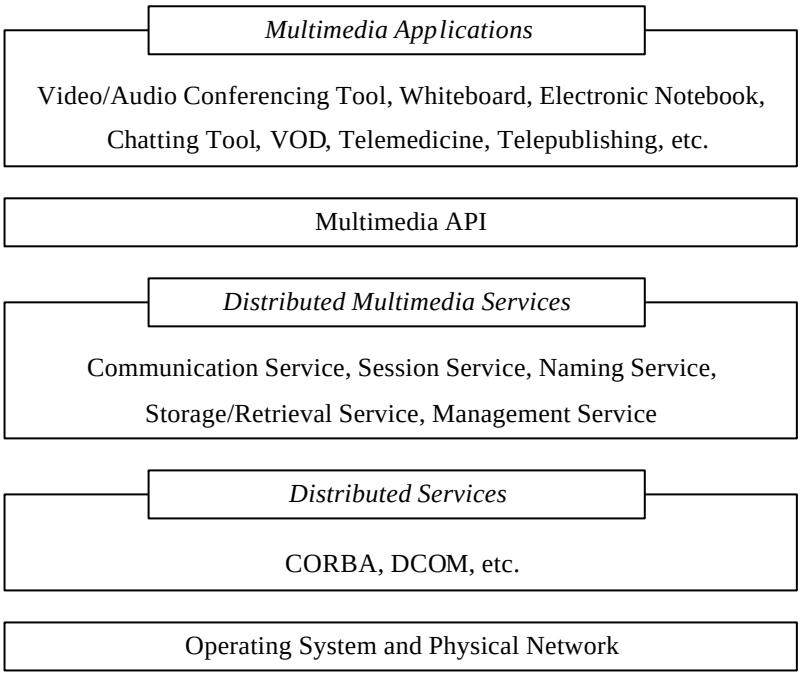


그림 2. MAESTRO 의 계층적 구조

응용 프로그램의 아래 층은 **멀티미디어 API** 를 제공한다. 이 API 층은 객체 지향 방법론을 이용해서 설계되었으며 여러 클래스 정의들로 구성된다. API 아래 층에는 **분산 멀티미디어 서비스**를 제공해 주는 여러 **분산 객체**들이 존재한다. 이 층의 분산 객체들이 제공해 주는 서비스들로 네임 서비스

(Name Service), 통신 서비스 (Communication Service), 세션 서비스 (Session Service), 저장/인출 서비스 (Storage/Retrieval Service)와 관리 서비스 (Management Service) 등이 있다. 이 층의 객체들은 CORBA [13, 14] 나 DCOM [16] 등과 같은 분산 객체 기술을 이용해서 형성되기 때문에 멀티미디어 응용 프로그램들은 이 객체들이 제공하는 여러 서비스들을 run-time 시에 사용하여 자신이 원하는 서비스를 찾거나 세션을 생성하거나 생성되어 있는 세션에 참가하거나 멀티미디어 데이터를 서로 주고 받거나 멀티미디어 데이터를 저장하고 인출하거나 시스템과 멀티미디어 응용 프로그램들을 모니터하고 제어할 수 있다.

분산 멀티미디어 서비스 층의 아래 층에는 분산 서비스를 가능하게 해주는 미들웨어가 존재하는데 분산 객체 기술에 해당되는 CORBA 나 DCOM 등의 기술이 이용될 수 있다. 가장 아래 층에는 다양한 운영체제 (Solaris, HP-UX, AIX, Digital Unix, IRIX, Linux, Windows NT 등)와 네트워크 기술 (ATM, FDDI, SONET, Fast Ethernet, Token Ring)이 존재할 수 있다.

위 구조에서 MAESTRO 를 구성하는 핵심은 멀티미디어 API 층과 분산 멀티미디어 서비스 층이다.

3.2. MAESTRO 멀티미디어 API

이 절에서는 MAESTRO 의 멀티미디어 API 층에 대해서 설명하는데 멀티미디어 API 를 정의하는 목적과 멀티미디어 API 를 구성하는 여러 객체들에 대해서 설명한다. 멀티미디어 API 를 정의하는 목적은 아래와 같다.

- ✍✍멀티미디어와 관련된 여러 클래스 정의들을 제공함으로써 멀티미디어 응용 프로그램들이 멀티미디어와 관련된 문제들을 쉽고 유연하게 사용할 수 있도록 한다.
- ✍✍멀티미디어 응용 프로그램 개발자들에게 공통된 인터페이스를 제공함으로써 새로운 멀티미디어 응용 프로그램들을 쉽게 개발할 수 있도록 한다.
- ✍✍여러 멀티미디어 응용 프로그램에서 중복적으로 사용되는 코드들을 멀티미디어 API 에 포함시킴으로써 코드의 재사용 정도를 높인다.

✍✍비디오, 오디오 등을 포함하여 다양한 미디어들을 처리하고 화면에 출력할 수 있도록 함으로써 멀티미디어 응용 프로그램들이 다양한 미디어들을 쉽게 다룰 수 있도록 한다.

✍✍멀티미디어 응용 프로그램의 개발 시간을 단축시킨다.

멀티미디어 API 에는 **미디어 클래스 계층** (Media Class Hierarchy), **컴포넌트 클래스 계층** (Component Class Hierarchy), **디바이스 클래스 계층** (Device Class Hierarchy)의 세 가지 클래스 계층이 정의되어 있고 아래 층의 분산 멀티미디어 서비스를 이용할 수 있도록 해 주는 클래스들도 정의되어 있다. 미디어 클래스 계층은 다양한 종류의 미디어들을 모델링하여 멀티미디어 응용 프로그램들이 여러 미디어들을 쉽게 다룰 수 있도록 해 준다. 컴포넌트 클래스 계층은 비디오와 오디오 같은 미디어를 계속해서 생산하거나 소비하는 **생산자** (Producer)와 **소비자** (Consumer)를 모델링하여 클래스로 정의하고 있다. 마지막으로 디바이스 클래스 계층은 카메라, 마이크, 출력 장치, 스피커와 같은 멀티미디어 관련 디바이스들을 모델링하여 클래스로 정의하고 있다. 그림 3, 4, 5는 각 클래스 계층 구조를 보여 준다.

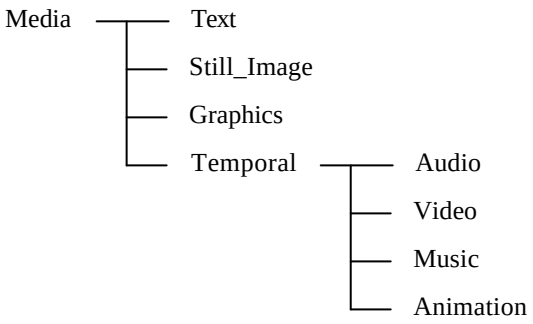


그림 3. Media Class Hierarchy

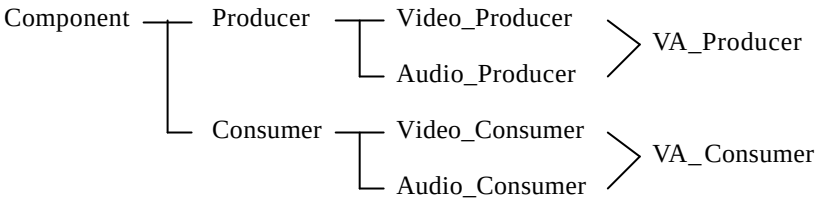


그림 4. Component Class Hierarchy

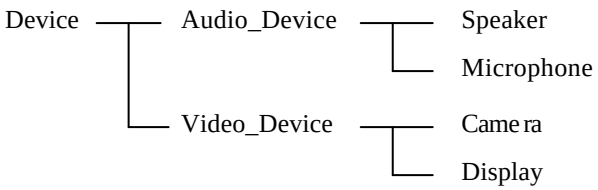


그림 5. Device Class Hierarchy

3.3. MAESTRO 분산 멀티미디어 서비스

이 절에서는 분산 멀티미디어 응용 프로그램들을 수행하고 관리하는데 중요한 역할을 차지하는 MAESTRO 의 분산 멀티미디어 서비스들에 대해서 설명한다. 현재 통신 서비스, 네임 서비스, 세션 서비스, 저장/인출 서비스, 관리 서비스의 다섯 가지 분산 멀티미디어 서비스가 정의되어 있다. 각 분산 멀티미디어 서비스는 분산 객체들로 구성되며 이 분산 객체들이 해당 서비스를 제공해 준다.

현재 다섯 가지 분산 멀티미디어 서비스가 정의되어 있으나 새로운 서비스가 필요하다면 이 서비스를 제공해 주는 분산 객체들을 쉽게 정의해서 추가할 수 있다. 즉, 새로운 서비스를 제공하는 분산 객체들은 뒤에서 설명할 네임 서비스를 이용해서 자신들을 등록하기 때문에 멀티미디어 응용 프로그램은 네임 서비스를 통해서 기존의 서비스들 뿐만 아니라 새롭게 등록되는 모든 서비스들을 이용할 수 있다. 이제 현재 정의되어 있는 각각의 분산 멀티미디어 서비스에 대해서 설명한다.

3.3.1. 통신 서비스 (Communication Service)

통신 서비스는 멀티미디어 응용 프로그램들이 다양한 종류의 미디어들을 서로 주고 받을 수 있도록 해 준다. 다양한 종류의 멀티미디어 응용 프로그램들을 지원하기 위해서 통신 서비스는 다자간 통신 (Multi-point Communication), 흐름 제어 (Flow Control), 신뢰성 있는 통신 (Reliable Communication), 대역폭 제어 (Bandwidth Control), 미디어 동기화 (Synchronization), 실시간 통신 (Real-time Communication) 등의 기능을 제공해 주어야 한다.

3.3.2. 네임 서비스 (Name Service)

분산 멀티미디어 서비스들을 제공해 주는 객체들은 네트워크 상에서 분산되어 있다. 이런 환경에서 멀티미디어 응용 프로그램들은 자신이 원하는 서비스를 제공해 주는 객체들을 쉽게 찾을 수 있어야 한다. 네임 서비스의 역할은 특정 객체에 특정 이름을 달아서 응용 프로그램들이 이름만을 가지고 특정 객체를 찾을 수 있도록 해 준다. 따라서 어떤 객체가 서비스를 제공하고자 할 때는 네임 서비스를 이용해서 자신의 이름과 자신을 등록해서 응용 프로그램들은 객체의 이름만을 가지고 해당 객체를 찾을 수 있다.

3.3.3. 세션 서비스 (Session Service)

만약 여러 사용자들이 공동 작업을 하고자 한다면 이 사용자들은 여러 종류의 응용 프로그램들을 동시에 사용하여 서로 공동 작업을 하게 된다. 이럴 경우 공동 작업을 하는 사용자들을 하나의 그룹으로 만들어서 각 사용자들에 대한 정보와 각 사용자들이 사용하는 응용 프로그램들의 정보를 관리하여야 한다. 세션 서비스는 이와 같이 공동 작업을 하고자 하는 사람들을 하나의 그룹으로 묶어주는 서비스이다. 즉 세션 서비스를 이용해서 하나의 세션을 생성하거나(creating) 원하는 세션을 찾거나(finding) 세션에 참여하거나(joining) 참여한 세션을 떠나거나(leaving) 생성한 세션을 없앨 수 있다(destroying).

3.3.4. 저장/인출 서비스 (Storage/Retrieval Service)

멀티미디어 API 층에서 다양한 종류의 미디어 객체들이 정의되었다. 저장/인출 서비스는 기본적으로 멀티미디어 API 층에서 정의된 여러 미디어 객체들을 저장하거나 저장되어 있는 미디어 객체들을 인출할 수 있도록 해 준다. 저장/인출 서비스는 이와 같은 기본적인 서비스 위에서 Video On Demand (VOD)나 디지털 라이브러리 서비스와 같은 보다 상위 레벨의 서비스들을 제공할 수 있다.

3.3.5. 관리 서비스 (Management Service)

관리 서비스 [5, 6, 7]는 MAESTRO 에서 분산 멀티미디어 서비스를 제공하는 분산 객체들과 응용 프로그램들을 관리하기 위해서 필요하다. 이 서비스

는 분산 객체들과 응용 프로그램들을 모니터하여 그 정보를 제공해 주거나 분산 객체들과 응용 프로그램들의 장애를 해결하거나 분산 객체들과 응용 프로그램들을 제어할 수 있는 기능을 제공한다. 관리 서비스는 분산 객체들이 보다 효율적이고 안정적으로 서비스를 제공할 수 있도록 하여 응용 프로그램들이 큰 문제 없이 수행될 수 있도록 한다.

4. MAESTRO 통신 서비스에 대한 요구 사항

통신 서비스에서는 다양한 종류의 멀티미디어 응용 프로그램들이 요구하는 여러 기능들을 제공해 주어야 하는데 이 장에서는 이러한 기능들에 대해서 설명한다. 아래는 여러 멀티미디어 응용 프로그램들의 통신 서비스에 대한 기능적 요구 사항들을 나열한 것이다.

✍✍다자간 통신 (Multi-point Communication)이 가능해야 한다: 다자간 통신은 화상 회의 프로그램이나 공동 작업을 위한 프로그램들의 개발에서 중요한 역할을 차지한다. 다자간 통신에서는 일대일 통신 (One-to-One Communication), 일대다 통신 (One-to-Mnay Communication), 다대일 통신 (Many-to-One Communication), 다대다 통신 (Many-to-Many Communication)이 가능해야 한다.

✍✍흐름 제어 (Flow Control)가 되어야 한다: 흐름 제어란 송신 프로그램이 데이터를 전송할 때 수신 프로그램은 송신 프로그램이 데이터를 보낸 순서대로 데이터를 받을 수 있도록 해 주는 것이다.

✍✍신뢰성 있는 통신 (Reliable Communication)이 보장되어야 한다: 패킷 전송 네트워크일 경우 네트워크 상의 부하 (load)가 심하면 전송되는 패킷이 소실될 수 있다. 이런 경우 재전송 등을 통해서 통신의 신뢰성을 보장해 주어야 한다.

✍✍멀티미디어 응용 프로그램들에게 대역폭 제어 (Bandwidth Control)의 기능을 제공해야 한다: 멀티미디어 응용 프로그램들은 통신 서비스를 이용해서 자신이 원하는 통신 대역폭을 요청할 수 있어야 하고 통신 서비스를 제공하는 분산 객체들은 응용 프로그램이 요청한 통신 대역폭을 응용 프로그램의 통신 시간 동안 보장해 주어야 한다.

✍✍여러 종류의 미디어를 여러 채널을 통해서 동시에 전송할 경우 각 미디어들을 동기화 (synchronization)시킬 수 있어야 한다: 화상 회의 프로그램의 경우 종종 비디오와 오디오를 동시에 전송해야 한다. 비디오와 오디오를 전송할 때 하나의 채널을 이용해서 전송할 수도 있으나 보다 효율적인 전송과 처리를 위해서 비디오와 오디오를 두

채널을 이용해서 전송하는 경우가 많다. 이럴 경우 두 채널을 시간에 맞게 동기화 시켜줄 수 있어야 한다.

✍✍실시간 통신 (Real-time Communication)이 가능해야 한다: 대화형 멀티미디어 응용 프로그램일 경우 멀티미디어의 실시간 전송이 관건이며 이런 종류의 응용 프로그램들을 위해서 필요에 따라서 실시간 통신을 보장해야 한다.

모든 멀티미디어 응용 프로그램들이 위에 열거된 모든 요구 사항들을 필요로 하는 것은 아니며 멀티미디어 응용 프로그램의 종류에 따라서 중요시되는 요구 사항들의 차이를 보이고 있다. 예를 들면, 화상 회의 프로그램의 경우 다자간 통신, 미디어 동기화, 실시간 통신 등이 중요하지만 신뢰성 있는 통신은 별로 중요하지 않다. 이에 반해 전자 공책이나 화이트 보드와 같은 응용 프로그램의 경우 다자간 통신과 신뢰성 있는 통신 등이 중요하지만 실시간 통신은 별로 중요하지 않다. 따라서 위의 모든 요구 사항을 만족시키기가 어렵다면 응용 프로그램의 동작에 중요한 요구 사항들을 만족시키고 그렇지 않은 요구 사항들은 어느 정도까지 희생시킬 수 있다.

그런데 통신 서비스에서 실제로 위에서 열거된 요구 사항들을 모두 만족시킨다는 것은 상당히 어려운 일이다. 이것은 여러 요구 사항들을 모두 만족시키려다 보니 새로운 문제가 발생하고 본질적으로 서로 상충되는 요구 사항들이 포함되어 있기 때문이다. 예를 들면, 지금까지는 클라이언트/서버 방식의 일대일 통신을 이용하는 프로그램들이 많았는데 멀티미디어 응용 프로그램들은 일대다 통신과 다대다 통신을 흔히 이용한다. 그런데 문제는 일대다 통신에서 대역폭 제어의 기능을 제공하는 것이 일대일 통신에서 대역폭 제어의 기능을 제공하는 것보다 더욱 어렵다는 것이다. 이것은 일대다 통신에 참여하는 응용 프로그램들이 서로 대역폭 (bandwidth)이 다르거나 교통량 (traffic)이 다른 네트워크 상에 존재할 수 있기 때문에 일정 수준의 대역폭을 유지하기가 힘들다. 또 다른 예는 신뢰성 있는 통신과 실시간 통신은 서로 상충되는 특성을 가지고 있기 때문에 이 두 가지를 동시에 만족 시키기가 힘들다는 것이다. 즉, 신뢰성 있는 통신을 보장하기 위해서는 경우에 따라서 데이터를 재전송할 필요가 있는데 이것은 데이터의 전송이 지연된다는 것을 의미하므로 실시간 통신을 저해시킨다. 그리고 실시간 전

송을 보장하기 위해서는 경우에 따라서 전송이 늦어지는 데이터를 버릴 필요가 있는데 이것은 신뢰성 있는 통신을 저해시킨다.

따라서 이 논문에서 제시하는 통신 서비스는 위에서 열거된 모든 요구 사항을 만족시키지는 못하며 다자간 통신, 흐름 제어, 신뢰성 있는 통신, 대역폭 제어 등의 기능을 제공한다.

5. MAESTRO 통신 서비스

2 장에서 응용 프로그램 수준에서의 멀티캐스팅과 네트워크 수준에서의 멀티캐스팅에 대한 여러 연구들에 대해서 설명하였다. 응용 프로그램 수준에서의 멀티캐스팅은 비효율적이기 때문에 멀티미디어 응용 프로그램의 운용에 바람직한 방법이 될 수 없다. 네트워크 수준에서의 멀티캐스팅이 바람직한 방법이지만 아직까지 많은 네트워크에서 멀티캐스팅을 지원하지 못하고 있고 설령 네트워크 수준에서 멀티캐스팅이 지원되더라도 흐름 제어, 신뢰성 있는 통신, 대역폭 제어, 미디어 동기화, 실시간 통신 등과 같은 다양한 기능들은 고려되지 않은 경우가 많다. 현재 네트워크 수준에서 멀티캐스팅을 지원하기 위한 많은 연구가 진행 중이기 때문에 앞으로 네트워크 수준에서 멀티캐스팅을 지원할 수 있는 표준이 마련될 것이다. 그러나 이 표준이 실제로 모든 네트워크 적용되기까지는 많은 시간이 걸릴 것이다.

이제 이 장에서는 현존하는 네트워크 기술을 그대로 수용하고 다자간 통신, 흐름 제어, 신뢰성 있는 통신, 대역폭 제어 등을 지원할 수 있는 통신 서비스에 대해서 설명한다. 이 통신 서비스는 밑에 있는 네트워크 기술을 그대로 이용하고 **시스템 수준**에서 분산 객체들을 이용하여 멀티캐스팅, 흐름 제어, 신뢰성 있는 통신, 대역폭 제어 등의 기능을 제공한다. 이 장에서는 통신 서비스의 개요에 대해서 먼저 설명하고 그 다음에 통신 서비스에서 정의되어 있는 분산 객체들에 대해서 설명한다. 그 다음에 통신 서비스에서 다자간 통신, 흐름 제어, 신뢰성 있는 통신, 대역폭 제어 등이 어떻게 해결되고 있는 지 순서대로 설명한다.

5.1. MAESTRO 통신 서비스 개요

통신 서비스에서 멀티미디어 응용 프로그램은 Port 라는 분산 객체를 이용해서 통신한다. 다른 응용 프로그램들과 통신하고자 하는 모든 응용 프로그램들은 Port 를 먼저 생성해야 하고 그런 뒤 다른 응용프로그램이 생성한 Port 와의 **연결 설정** (Connection Establishment)을 통해서 **연결 통로** (Connection Passage)를 만든 후 통신한다. 그림 6 은 두 Port 사이의 연결

통로를 보여 주고 있는데 응용 프로그램이 Port 의 인터페이스를 호출하여 데이터를 Port 에게 전달하면 그 Port 는 다른 Port 와의 연결 통로 (Connection Passage)를 통해서 데이터를 보내 준다. MAESTRO 통신 서비스는 Port 와 Port 사이의 연결 통로를 만들기 위해서 밑에 있는 네트워크 기술을 그대로 이용한다. 즉, 연결 통로를 만들기 위해서 TCP 나 UDP 와 같은 네트워크 상에서 제공되는 유니캐스트 기술을 그대로 이용할 수 있으며 원한다면 멀티캐스트 기술을 이용할 수도 있다. 아래 그림은 두 Port 사이의 연결 통로만을 보여 주고 있지만 여러 Port 들 사이의 멀티캐스팅을 위해서 Port 와 Port 가 서로 연결 통로로 연결되어 계층적 트리를 형성한다. Port 들이 계층적 트리를 형성하는 과정은 뒤에서 자세히 설명된다.

그림 6. 두 Port 사이의 연결 설정

MAESTRO 에서 도메인이란 분산 멀티미디어 서비스들이 멀티미디어 응용 프로그램들에게 제공되는 영역으로 정의되는데 도메인의 개념은 분산 멀티미디어 서비스들의 관리를 쉽게 하고 확장성 (scalabilty)을 높이기 위해서 사용된다. 만약 두 멀티미디어 응용 프로그램이 똑같은 도메인에 존재한다면 이 두 멀티미디어 응용 프로그램들은 통신 서비스를 이용해서 당연히 서로 통신할 수 있어야 한다. 그러나 두 멀티미디어 응용 프로그램이 서로 다른 도메인에 존재하더라도 이들 두 멀티미디어 응용 프로그램들은 통신 서비스를 이용해서 서로 통신할 수 있어야 한다. MAESTRO 통신 서비스에서 전자를 도메인 내 통신 (Intra-domain Communication)이라 부르고 후자를 도메인 간 통신 (Inter-domain Communication)이라 부른다. 도메인 내 통신과 도메인 간 통신의 개념은 그림 7 에 잘 나타나 있다.

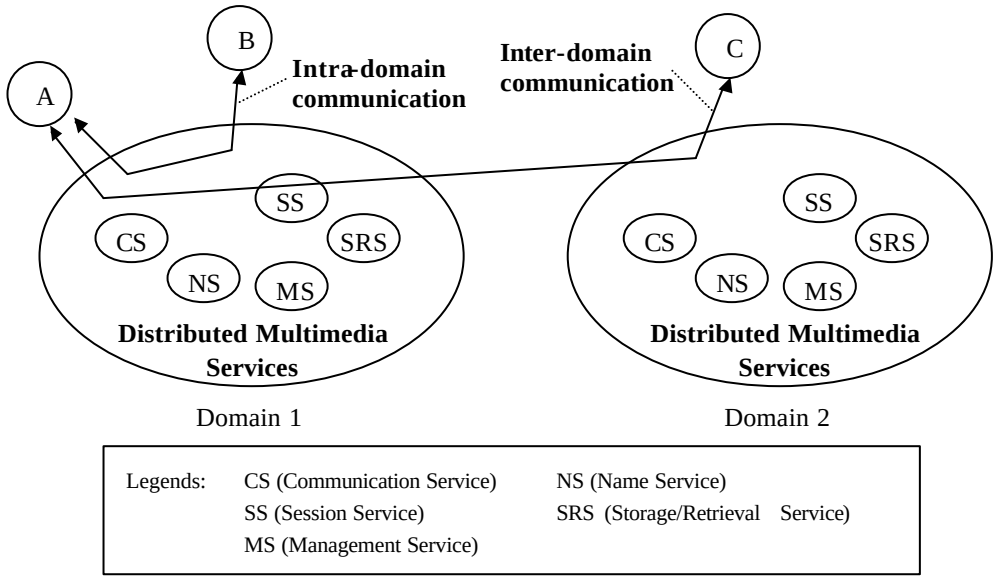


그림 7. 도메인 내 통신과 도메인 간 통신

그림에서 멀티미디어 응용 프로그램 A 와 B 는 똑같은 도메인이 제공하는 분산 멀티미디어 서비스들을 이용하고 있으며 이들의 통신은 도메인 내 통신에 해당된다. 그림에서 A 와 C 는 서로 다른 도메인이 제공하는 분산 멀티미디어 서비스들을 이용하고 있으며 이들의 통신은 도메인 간 통신에 해당된다.

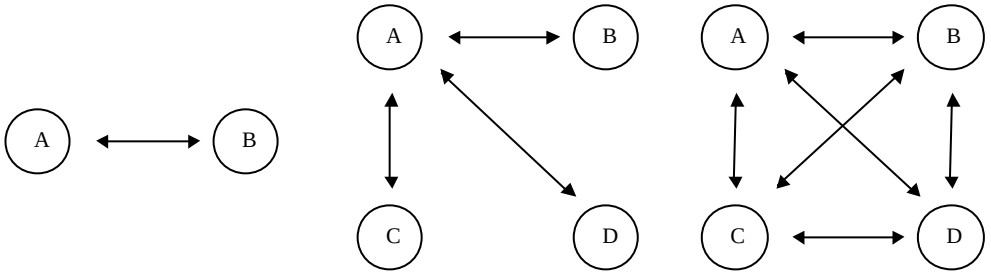
5.2. MAESTRO 통신 서비스를 구성하는 분산 객체들

통신 서비스에서 사용되는 분산 객체들로 CommunicationFactory, ConnectionManager, Port, Connector 와 Channel 객체들이 있다. CommunicationFactory 의 역할은 멀티미디어 응용 프로그램들이 서로 통신할 때 사용되는 Port, Connector 와 Channel 객체들을 생성하고 관리하는 것이다. 만약 어떤 멀티미디어 응용 프로그램이 통신하고자 한다면 CommunicationFactory 를 이용해서 Port 를 먼저 생성해야 한다. 그런 뒤 이 응용 프로그램은 역시 자신의 Port 를 생성한 다른 응용 프로그램과 서로 통신할 수 있다. 이와 같이 멀티미디어 응용 프로그램들은 자신의 Port 를 생성하고 생성된 Port 를 통해서 서로 통신하는데 하나의 Port 가 다른 Port 와 통신하기 위해서는 먼저 **연결 설정** (Connection Establishment)이 이루어져

야 한다. 만약 어떤 Port 가 다른 Port 와의 연결 설정을 원한다면 **연결 요청** (Connection Request)을 ConnectionManager 에게 전달해야 한다. Port 들의 실제적인 연결 설정은 모두 ConnectionManager 에 의해서 이루어진다. 다자간 통신은 여러 Port 들 사이에서 여러 연결 설정들이 이루어 진 후 가능하다. Connector 는 **도메인 간 통신** (Inter-domain Communication)이 필요할 때 사용되는데 Connector 의 역할에 대해서는 뒤에서 자세히 설명될 것이다. Channel 은 다자간 통신에 참여하고 있는 모든 Port 들의 레퍼런스(reference)들을 관리하기 위해서 사용된다.

5.3. 다자간 통신

다자간 통신 (일대일 통신, 일대다 통신, 다대일 통신, 다대다 통신)은 멀티미디어 응용 프로그램의 개발에 있어서 아주 중요한 역할을 차지한다. 다자간 통신에 대해서 자세히 설명하기 전에 먼저 일대일, 일대다, 다대일, 다대다 통신이 각각 무엇을 의미하는 지 설명한다.



(a) 일대일 통신 (b) 일대다 또는 다대일 통신 (c) 다대다 통신

그림 8. 일대일, 일대다, 다대일, 다대다 통신

그림 8 의 (a)는 일대일 통신의 예를 보여 주는데 A 와 B 가 서로 데이터를 주고 받을 수 있다. 그림 8 의 (b)는 일대다 또는 다대일 통신의 예를 보여 주는데 A 가 데이터를 보내면 B 와 C 와 D 가 받고 B 나 C 나 D 가 데이터를 보내면 A 가 받게 된다. 그림 8 의 (c)는 다대다 통신의 예를 보여 주는데 한 노드 (node)에서 데이터를 보내면 다른 모든 노드들이 데이터를 받게 된다. 예를 들어 A 가 데이터를 보내면 B 와 C 와 D 가 모두 데이터를 받게 되고 B 가 데이터를 보내면 A 와 C 와 D 가 모두 데이터를 받게 된다.

MAESTRO 통신 서비스는 일대다 통신과 다대다 통신만을 지원하는데 일대일 통신은 일대다 통신이나 다대다 통신을 이용해서 가능하고 다대일 통신은 일대다 통신을 이용해서 가능하다. MAESTRO 통신 서비스에서 일대다 통신과 다대다 통신을 따로 지원하기 위해서 두 종류의 Port 를 이용하는데 하나는 **일대다 Port** 이고 다른 하나는 **다대다 Port** 이다. 따라서 일대다 또는 다대일 통신을 원하는 멀티미디어 응용 프로그램들은 일대다 Port 를 생성해서 서로 통신하여야 하고 다대다 통신을 원하는 멀티미디어 응용 프로그램들은 다대다 Port 를 생성해서 서로 통신해야 한다. 멀티미디어 응용 프로그램은 Port 를 생성할 때 일대다 Port 를 생성할 것인지 다대다 Port 를 생성할 것인지를 지정할 수 있다. 그리고 일대다 Port 는 일대다 Port 들끼리 연결되어 이용되고 다대다 Port 는 다대다 Port 들끼리 연결되어 이용되기 때문에 일대다 Port 와 다대다 Port 를 연결해서 이용할 수는 없다.

MAESTRO 통신 서비스는 다자간 통신을 위해서 Port 와 Channel 객체들을 이용한다. 그림 9 는 다자간 통신이 개념적으로 어떻게 이루어지는가를 보여준다.

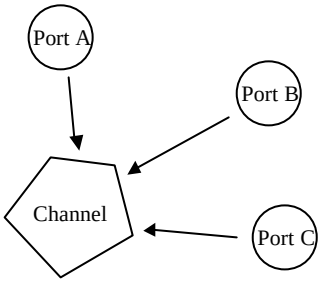


그림 9. 다자간 통신에서 Port 와 Channel

그림에서 Port A 를 생성한 멀티미디어 응용 프로그램은 Port A 의 인터페이스를 이용해서 Channel 을 생성한다. Port A 는 Channel 을 생성하는 것과 동시에 생성된 Channel 에 참여한다. 그런 후 Port B 와 Port C 는 Port A 로의 연결 요청을 하고 그 결과 Port A 가 생성한 Channel 에 참여하게 된다. 따라서 하나의 Port 가 Channel 에 참여할 수 있는 방법은 두 가지가 있는데

✍✍첫째는 Port 가 직접 Channel 을 생성하거나

✍✍둘째는 Port 가 이미 Channel 에 참여하고 있는 다른 Port 로의 연결 요청을 통해서 가능하다.

통신 서비스에서 다자간 통신을 위해서 Port 들이 서로 연결되어 계층적 트리를 만든다. 그림 10 은 Port 들의 계층적 트리가 어떻게 형성되는가를 보여준다. 그림에서 Port A 는 먼저 Channel 을 생성함으로써 자신을 Channel 에 참여시키고 다른 Port 들이 Port A 로 연결할 수 있도록 한다. Port B 는 Port A 가 이미 Channel 에 참여하고 있으므로 Port A 로 연결할 수 있고 마찬가지로 Port C 와 D 도 Port A 로 연결할 수 있다. 연결 후 Port B 와 C 는 모두 Port A 가 생성한 Channel 에 자동으로 참여하게 된다. 이제 Port D 는 Port A 가 처음에 생성한 Channel 에 참여하고 있기 때문에 Port E 와 Port F 는 Port D 로 연결해서 그 Channel 에 참여할 수 있다. 멀티미디어 응용 프로그램들은 위와 같이 Port 들을 계층적 트리 형태로 연결해서 일대다 또는 다대다 통신을 할 수 있다.

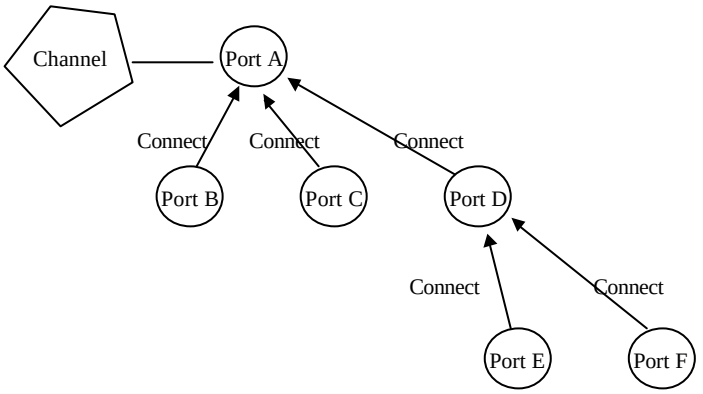


그림 10. 다자간 통신에서의 계층적 트리

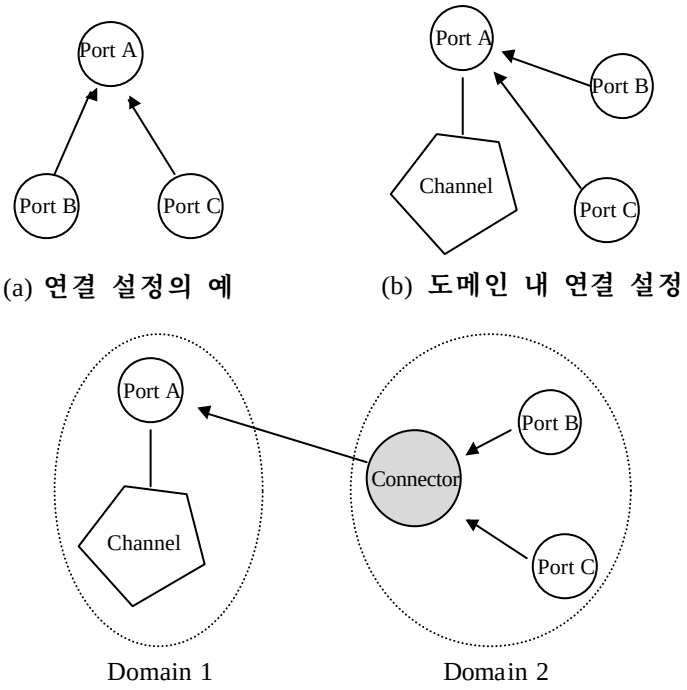
일대다 통신의 경우 일대다 Port 를 이용하고 다대다 통신에서는 다대다 Port 를 이용하지만 두 통신에서 계층적 트리를 만드는 과정은 앞에서 설명한 방법과 동일하다. 그러나 일대다 통신과 다대다 통신에서 계층적 트리가 형성된 후 계층적 트리에서 데이터가 흐르는 방법은 서로 다르다. 이제 일대다 통신과 다대다 통신에서 데이터가 흐르는 방법을 그림 10 을 이용해서 순서대로 설명한다.

일대다 통신에서 루트 Port 가 데이터를 보내면 다른 모든 Port 들이 그

데이터를 받고 루트 Port 가 아닌 Port 가 데이터를 보내면 루트 Port 만 데이터를 받게 된다. 예를 들어 멀티미디어 응용 프로그램이 Port A 의 인터페이스를 호출해서 Port A 로 데이터를 보내면 Port A 는 데이터를 자식 Port 들인 Port B, C 와 D 에게 보낸다. 부모 Port 로부터 데이터를 받은 Port D 는 자신이 데이터를 가지는 동시에 자식 Port 들인 Port E 와 F 에게 데이터를 보낸다. 다른 예로 만약 멀티미디어 응용 프로그램에 Port F 의 인터페이스를 호출해서 데이터를 보내면 Port F 는 Port D 에게 데이터를 보내고 자식 Port 로부터 데이터를 받은 Port D 는 자신이 데이터를 가지지 않고 단지 부모 Port 인 Port A 에게 데이터를 전달한다.

다대다 통신의 경우 계층적 트리 내의 어떤 Port 가 데이터를 보내면 다른 모든 Port 들이 그 데이터를 받게 된다. 예를 들어 멀티미디어 응용 프로그램이 루트 Port 인 Port A 로 데이터를 보내면 이 데이터는 일대다 통신에서 데이터가 흐르는 것과 똑같이 흐른다. 만약 멀티미디어 응용 프로그램이 Port F 로 데이터를 보내면 Port F 는 부모 Port 인 Port D 에게 데이터를 보낸다. Port D 는 자신이 데이터를 가지는 것과 동시에 데이터를 자식 Port 인 Port E 와 부모 Port 인 Port A 에게 보내 준다. 데이터를 받은 Port A 는 Port D 와 마찬가지로 자신이 데이터를 가지는 것과 동시에 자식 Port 들인 Port B 와 C 에게 데이터를 보내 준다.

이제 통신 서비스의 도메인 내 통신과 도메인 간 통신에서 분산 객체들이 구체적으로 어떻게 이용되는가를 설명한다. 예를 들어 그림 11 의 (a)와 같은 Port 들의 계층적 트리가 만들어졌다고 가정하자. 만약 모든 Port 들이 지역 (local) 도메인에 위치한다면 Port 들의 연결은 그림 11 의 (b)와 같이 모두 Port 들만으로 이루어진다. 그러나 만약 Port A 가 지역 도메인에 위치하고 Port B 와 C 는 원격 (remote) 도메인에 위치한다면 그림 11 의 (c)와 같이 원격 도메인 내에 Connector 가 생성되어 Port 들의 연결에 이 Connector 가 이용된다.



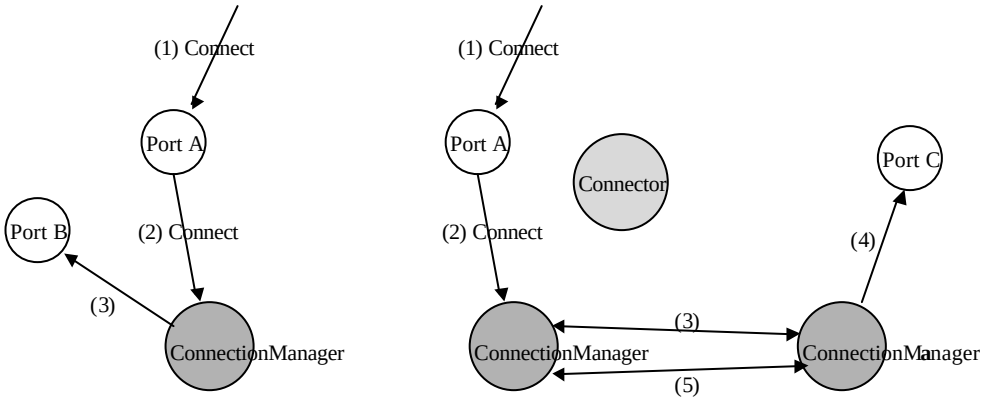
(a) 연결 설정의 예 (b) 도메인 내 연결 설정

(c) 도메인 간 연결 설정

그림 11. 도메인 내 연결 설정과 도메인 간 연결 설정

비록 도메인 간 통신에서 Connector 를 이용하지 않고 도메인 내 통신에서처럼 Port 들만으로 연결할 수도 있으나 도메인 간 통신에서 Connector 를 이용하는 이유는 도메인 간 통신을 보다 효율적으로 하기 위해서 이다. 예를 들어 만약 데이터가 도메인을 건너서 전달되어야 한다면 데이터의 한 카피만 Connector 에게 전달되고 Connector 는 자신이 위치한 도메인의 해당 Port 들에게 데이터를 전달함으로써 도메인 간에는 한 카피의 데이터만 전달되도록 한다.

이제 이 논문에서는 Port 들의 연결 설정 과정을 구체적으로 설명한다. 먼저 두 Port 가 똑같은 도메인 내에 존재할 경우인 도메인 내 연결 설정 과정을 설명하고 그 다음에 두 Port 가 서로 다른 도메인 내에 존재할 경우인 도메인 간 연결 설정 과정을 설명한다.



(a) 도메인 내 연결 설정 (b) 도메인 간 연결 설정

그림 12. 연결 설정 과정

도메인 내 연결 설정 과정은 그림 12 의 (a)에 나타나 있다. 아래는 도메인 내 연결 설정 과정을 순서대로 기술한 것이다.

(1) 멀티미디어 응용 프로그램은 Port A 에게 Port B 로의 연결을 요청한다. 이 때 멀티미디어 응용 프로그램은 Port B 가 위치하는 도메인 이름과 Port B 의 Port 번호를 인자 (argument)로 넘겨준다. Port 번호는 Port 가 생성될 때 부여되는 값으로 각 Port 는 도메인 내에서 유일한 Port 번호를 가진다. 따라서 도메인 이름과 Port 번호를 이용해서 특정 Port 를 지정할 수 있다.

(2) Port A 는 응용 프로그램으로부터의 연결 요청을 ConnectionManager 에게 넘긴다.

(3) ConnectionManager 는 도메인 이름과 Port 번호를 이용해서 해당 Port 를 찾고 두 Port 사이의 트리 관계를 형성한다.

도메인 간 연결 설정 과정은 그림 12 의 (b)에 나타나 있다. 아래는 도메인 간 연결 설정 과정을 순서대로 기술한 것이다.

(1) 멀티미디어 응용 프로그램은 Port A 에게 Port C 로의 연결을 요청한다.

(2) Port A 는 응용 프로그램으로부터의 연결 요청을 ConnectionManager 에게 넘긴다.

(3) ConnectionManager는 해당 Port가 원격 도메인에 존재한다는 것을 발견하고 원격 도메인의 ConnectionManager에게 Port들의 연결 설정을 도와 줄 것을 요구한다.

(4) 원격 ConnectionManager는 해당 Port를 찾는다.

(5) 지역 ConnectionManager와 원격 ConnectionManager는 서로 협력하여 필요한 정보를 전달하고 Connector를 생성한 후 이 Connector를 이용하여 두 Port 사이의 트리 관계를 형성한다.

지금까지 통신 서비스에서 다자간 통신과 관련된 여러 개념들과 객체들의 동작 방식에 대해서 설명하였다. 이제 위와 같은 다자간 통신 구조 하에서 흐름 제어, 신뢰성 있는 통신, 대역폭 제어 등은 어떻게 해결되는가를 순서대로 설명한다.

5.4. 흐름 제어와 신뢰성 있는 통신

그림 13 와 같은 계층적 트리에서 Port 들은 **연결 통로** (Connection Passage)에 의해 서로 연결되어 있는데 전체 트리에서의 흐름 제어와 신뢰성 있는 통신의 보장은 각 연결 통로에서의 흐름 제어와 신뢰성 있는 통신의 보장을 통해서 가능하다.

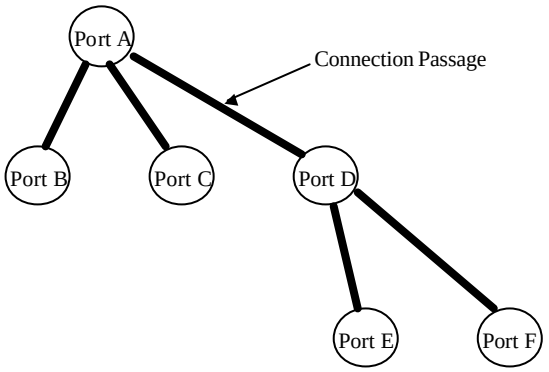


그림 13. 연결 통로에 의해 연결되어 있는 Port 들

각 연결 통로에서의 흐름 제어와 신뢰성 있는 통신의 보장 방법은 어떤 네트워크 기술을 이용하느냐에 따라서 달라진다. 예를 들어 TCP 를 이용해서 연결 통로를 만들 경우 연결 통로에서의 흐름 제어와 신뢰성 있는 통신은 TCP 자체에 의해서 보장되기 때문에 Port 들은 연결 통로에서의 흐름 제어와

신뢰성 있는 통신을 고려할 필요가 없다. 그러나 만약 UDP 를 이용해서 연결 통로를 만들 경우 연결 통로에서의 흐름 제어와 신뢰성 있는 통신이 보장되지 않으므로 Port 들은 연결 통로에서의 흐름 제어와 신뢰성 있는 통신을 보장할 수 있도록 동작하여야 한다.

신뢰성 있는 통신과 관련된 한 가지 문제점은 특정 연결 통로의 교통량이 갑자기 증가하거나 네트워크 상의 장애에 의해서 특정 연결 통로가 끊어질 경우 이것이 전체 트리에서의 통신에 영향을 미친다는 것이다. 예를 들어 그림 13 에서 현재 Port A 와 D 사이에 네트워크 상의 장애로 인해서 데이터가 올바르게 전송될 수 없다고 가정하자. 응용 프로그램이 Port A 로 데이터를 보냈을 경우 Port A 는 Port B, C 와 D 에 데이터를 보내야 한다. Port A 는 Port B 와 C 에게 별다른 문제 없이 데이터를 보낼 수 있으나 Port A 는 Port D 에게 데이터를 보낼 수가 없다. 전체 트리에서의 신뢰성 있는 통신을 보장하기 위해서 Port A 와 D 사이의 신뢰성 있는 통신이 보장되어야 하기 때문에 Port A 는 Port D 에게 데이터를 보낼 수 있을 때까지 계속해서 시도를 해야 한다. 그리고 이것은 전체 트리에서의 데이터 전송을 지연시킨다.

위와 같은 문제점을 방지하기 위해서 연결 통로에 Tmax 시간을 설정하여 Tmax 시간 동안 시도하여 데이터를 전송할 수 없을 경우 계층적 트리에서 해당 연결 통로를 제거하여 트리를 재형성함으로써 전체 트리에서 데이터 전송의 지연이 장시간 발생하지 않도록 한다.

5.5. 대역폭 제어

통신 서비스에서 대역폭 제어 (Bandwidth Control)의 기능을 제공해 준다는 말은 멀티미디어 응용 프로그램들이 통신 서비스를 이용해서 자신이 원하는 통신 대역폭을 요청할 수 있고 응용 프로그램의 통신 시간 동안 그 통신 대역폭이 보장된다는 것을 의미한다.

멀티미디어 응용 프로그램은 자신이 생성한 Port 를 다른 Port 로 연결시키기 위해서 자신이 생성한 Port 의 연결 요청 인터페이스를 호출하는데 이때 인자로 도메인 이름과 Port 번호와 함께 자신이 원하는 통신 대역폭 값을 넘겨 준다. 그러면 연결 설정 과정에서 응용 프로그램이 요청한 통신 대역폭이 제공될 수 있는가를 조사하여 그렇다면 연결 설정 과정을 성공적으

로 종료하고 그렇지 않다면 응용 프로그램이 요청한 통신 대역폭이 제공될 수 없음을 알려 준다.

일단 이렇게 연결 설정이 이루어지면 통신 서비스는 응용 프로그램의 통신 시간 동안 통신 대역폭을 보장해 주어야 한다. 만약 밑에 있는 네트워크 기술이 통신 대역폭 보장과 관련된 기능을 제공한다면 별 어려움 없이 통신 대역폭을 보장해 줄 수 있다. 그러나 그렇지 못할 경우 통신 서비스가 자체적으로 통신 대역폭을 보장해 주기 위해서 노력하여야 한다.

그런데 패킷 통신망일 경우 네트워크가 자체적으로 통신 대역폭을 보장해 주지 못하기 때문에 응용 프로그램의 통신 중에 통신 대역폭이 응용 프로그램이 요청한 값 아래로 떨어질 수 있으며 따라서 통신 서비스는 통신 대역폭을 완전하게 보장해 줄 수 없다.

MAESTRO 통신 서비스는 각 연결 통로의 통신 대역폭이 보장되고 있는가를 계속해서 모니터하여 통신 대역폭이 보장되지 못할 경우 계층적 트리에서 통신 대역폭을 보장해 줄 수 없는 연결 통로를 제거하여 트리를 재형성함으로써 전체 트리의 통신 대역폭을 유지한다.

6. MAESTRO 통신 서비스의 구현 및 이용

앞 장에서는 통신 서비스를 구성하는 여러 분산 객체들의 기능과 상호간의 동작 방식에 대해서 설명하였다. 이 장에서는 통신 서비스를 구성하는 분산 객체들이 어떻게 구현되었는가를 구체적으로 설명한다. 아울러 이 장에서는 멀티미디어 응용 프로그램이 통신 서비스를 실제로 어떻게 이용할 수 있는가에 대해서도 설명한다.

6.1. MAESTRO 통신 서비스의 구현

통신 서비스를 구현하기 위한 하드웨어 플랫폼으로 Solaris 2.4 를 운영 체재로 사용하는 Sun Sparc 20 과 Solaris 2.5 를 운영 제재로 사용하는 Sun Ultra 1 을 이용하였다. 카메라로부터 비디오 데이터를 읽어 들이기 위한 장치로 Sun Video Card [30]를 이용하였으며 Sun Video Card 를 이용할 수 있게 해 주는 Solaris XIL 라이브러리 [31]를 이용하였다.

통신 서비스를 구성하는 분산 객체들은 CORBA 와 C++ 언어를 이용해서 구현되었다. 각 분산 객체들의 인터페이스는 CORBA IDL 을 이용하여 정의되었으며 이것은 Appendix A 에 나타나 있다. 그리고 CORBA 제품으로 현재 가장 널리 사용되고 있는 IONA 의 Orbix 2.2 [28]를 이용하였다. Orbix 2.2 에 포함되어 있는 IDL 컴파일러를 이용하여 Appendix A 에 정의되어 있는 IDL 파일을 컴파일하면 C++ 소스 코드를 생성해 주는데 이 코드는 통신 서비스를 구성하는 분산 객체들의 뼈대에 해당되는 코드이다. 분산 객체들은 뼈대에 해당되는 이 C++ 코드의 내용을 채워서 구현되었다.

6.2. MAESTRO 통신 서비스의 이용

이 절에서는 통신 서비스에서 분산 객체들이 어떻게 이용되는가를 설명한다. 다음의 코드는 통신 서비스를 이용하여 데이터를 멀티캐스트하고자 하는 서버측의 주요 코드에 해당된다.

```
/*
 * cf means a reference to CommunicationFactory
 */

/* Create a port which supports one_to_many communication */
port_a = cf->create_port(Communication::one_to_many);

port_a->set_port_number(454);

channel_a = port_a->create_channel();

/* wait until another port joins the channel */
while (channel_a->port_count() == 1)
    ;

while (1) {

    /*
     * create media m
     */

    port_a->send_media(m);

  /* If all other ports have leaved the channel, stop sending media */
    if (channel_a->port_count() == 1)
        break;
}

port_a->destroy_channel(channel_a);
cf->destroy_port(port_a);
```

위 코드에서 응용 프로그램은 먼저 Port 를 생성하고 Port 번호를 454 번으로 설정한다. 그런 뒤 Channel 을 생성하고 다른 응용 프로그램들이 멀티캐스트 그룹에 참여하기를 기다린다. 이 응용 프로그램은 다른 응용 프로그램들이 멀티캐스트 그룹에 참여하면 Port 를 통하여 데이터를 전송하기 시작한다. 데이터의 전송은 다른 응용 프로그램들이 모두 멀티캐스트 그룹을 떠나면 루프를 빠져 나옴으로써 중지된다. 마지막으로 이 응용 프로그램은 생성된 Channel 과 Port 를 없앤다.

아래의 코드는 멀티캐스트 그룹에 참여하는 클라이언트 측의 주요 코드를 나타낸 것이다. 이 응용 프로그램은 앞에서와 마찬가지로 Port 를 생성하고 지역 도메인에서 454 번의 Port 번호를 가지는 Port 로 연결 요청을 한다. 연결 설정이 이루어진 후 이 응용 프로그램은 서버측이 보내 주는 데이터를 받아서 처리한다. 루프를 빠져 나온 후 연결 설정을 해제하고 생성된 Port 를 없앤 후 프로그램을 종료한다.

```
/*
 * cf means a reference to CommunicationFactory
  */

int      i;

port_a = cf->create_port(Communication::one_to_many);

/* Connect to a port which is located in "local" domain and whose port number is
 454 */
port_a->connect("local", 454, qos_p);

for (i=0; i<10; i++) {

    m  =  port_a->return_media();

   /* process media m */
}

port_a->disconnect();
cf->destroy_port(port_a);
```

7. MAESTRO 통신 서비스를 이용하는 멀티미디어 응용 프로그램

그림 14 는 통신 서비스를 이용해서 개발된 화상 회의 도구를 보여 준다. 이 도구는 X Window 상에서 C++ 언어를 이용하여 개발되었다. 사용자 인터페이스를 위해서 Motif 라이브러리를 이용하고 카메라로부터 비디오를 읽어들이기 위해 Sun Video Card 를 이용하였다. 사용자는 그림 14 의 왼쪽 윈도우에 나타나 있는 비디오 제어 (Video Control) 버튼들과 오디오 제어 (Audio Control) 버튼들을 이용해서 비디오 데이터와 오디오 데이터를 보내거나 받을 수 있다. 비디오 제어와 오디오 제어가 분리되어 있기 때문에 사용자는 이 도구를 비디오 회의 만을 위해 사용하거나 오디오 회의 만을 위해 사용할 수도 있다.

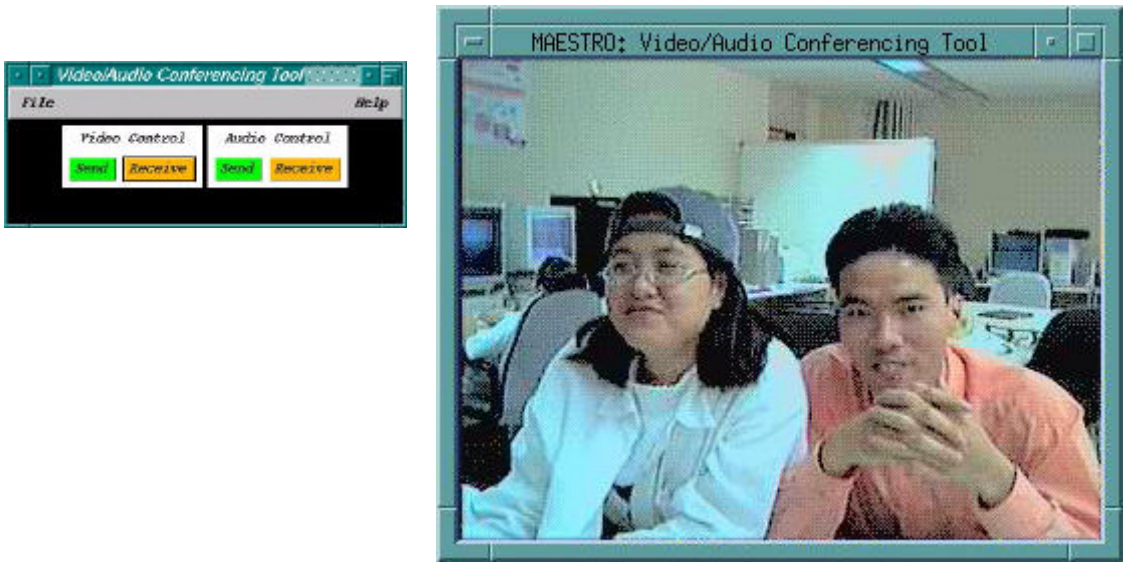


그림 14. 화상 회의 도구

그림 15 는 통신 서비스를 이용해서 개발된 비디오/오디오 멀티캐스팅 도구를 보여 주고 있다. 이 도구는 비디오 테이프를 상영하거나 저장되어 있는 비디오/오디오 데이터를 멀티캐스트할 수 있는 도구이다. 이 도구는 서버와 클라이언트로 구성되어 있는데 서버는 비디오/오디오 데이터를 멀티캐스트 해 주고 클라이언트는 멀티캐스트되는 비디오/오디오 데이터를 받아서

보여 준다. 그림 15 에서 서버는 amazon 이라는 이름의 도메인에 존재하며 Port 번호 200 번을 이용하여 Star Wars 라는 영화의 비디오/오디오 데이터를 멀티캐스트하고 있다. 사용자는 클라이언트 GUI 의 "Location" 필드에 도메인 이름과 Port 번호를 지정해서 자신이 원하는 서버에 접속할 수 있다. 화상 회의 도구와 마찬가지로 이 도구도 X Window 상에서 C++ 언어를 이용하고 Motif 라이브러리와 Sun Video Card 를 이용해서 개발되었다.

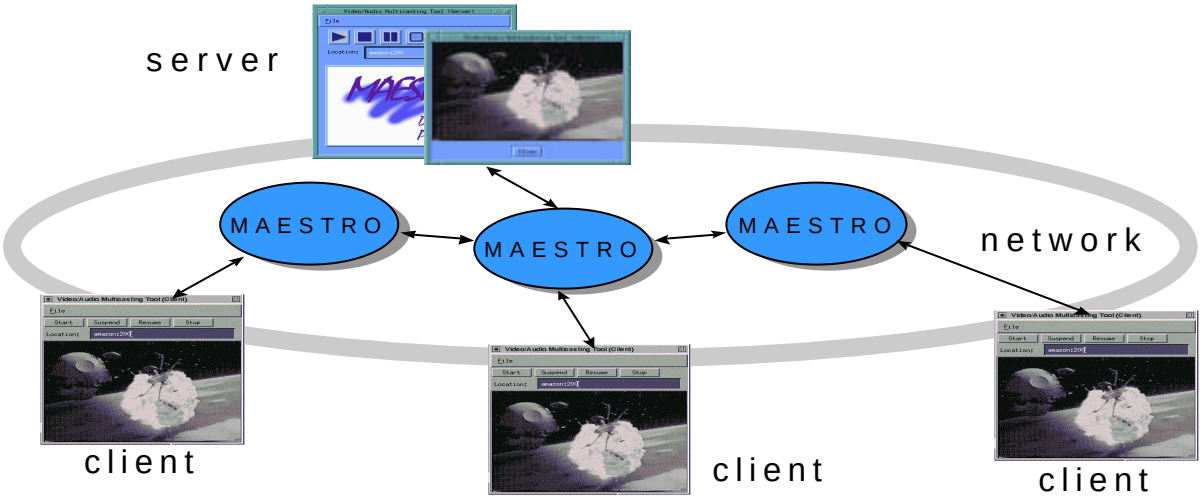


그림 15. 비디오/오디오 멀티캐스팅 도구

8. 결론 및 향후 계획

현재 비디오/오디오 데이터의 멀티캐스팅을 위해서 인터넷에서 MBONE 기술이 사용되고 있다. 그러나 MBONE 은 단순히 비디오/오디오 데이터를 포함하는 패킷을 멀티캐스트 해 줄 뿐이고 흐름 제어, 신뢰성 있는 통신, 대역폭 제어, 미디어 동기화, 실시간 통신 등과 같은 기능들을 효과적으로 지원하지 못한다. 현재 네트워크 수준에서 멀티캐스팅, 흐름 제어, 신뢰성 있는 통신, 대역폭 제어, 미디어 동기화, 실시간 통신 등의 기능들을 제공해 주기 위한 많은 연구가 진행 중이기 때문에 앞으로 네트워크 수준에서 이런 기능들을 지원할 수 있는 표준이 마련될 것이다. 그러나 표준이 마련되더라도 이 표준이 실제로 모든 네트워크에 적용되기까지 또 많은 시간이 걸릴 것이다.

따라서 이 논문에서는 현존하는 네트워크 기술을 그대로 수용하고 시스템 차원에서 분산 객체 기술을 이용하여 멀티미디어 응용 프로그램들에게 효율적이고 안정적인 통신 수단을 제공해 줄 수 있는 MAESTRO 통신 서비스에 대해서 설명하였다. MAESTRO 통신 서비스는 현재 멀티캐스팅, 흐름 제어, 신뢰성 있는 통신, 대역폭 제어의 기능들을 제공하고 있다.

MAESTRO 통신 서비스는 CORBA 객체들로 구현되었고 CORBA Object Request Broker 를 통해서 분산되었다. 가장 최근의 CORBA 표준 문서인 CORBA 2.0 표준 문서에는 아직 멀티미디어 응용 프로그램들을 위한 통신 수단을 제공하지 않는다. 이러한 문제를 인식하고 Object Management Group (OMG)은 비디오/오디오 서비스를 위한 RFP (Request for Proposal)를 내 놓았었다 [27]. 이 논문에서 제시하는 MAESTRO 통신 서비스가 CORBA 상에서 비디오/오디오 서비스를 정의하는데 기여하기를 바란다.

MAESTRO 통신 서비스가 효율적인 통신 서비스를 제공한다는 것을 보여주기 위해서 MAESTRO 통신 서비스의 구현과 아울러 통신 서비스를 이용한 비디오/오디오 멀티캐스팅 도구와 화이트보드를 개발하였다. 이 프로그램들은 MAESTRO 통신 서비스를 이용해서 쉽게 개발될 수 있었으며 효율적이고 안정적으로 운용될 수 있었다.

그러나 이 논문에서 제시하는 통신 서비스는 아직 좀 더 보완이 필요한데

미디어 동기화나 실시간 통신 등을 제공할 수 있는 방안에 대한 연구가 필요하다. 뿐만 아니라 통신 서비스가 효율적인 통신 수단을 제공해 준다는 것을 보여 주기 위해서 실험적인 수치를 산출하여 검증할 필요가 있다.

이 논문에서는 현재의 컴퓨팅 환경과 네트워킹 기술 상에서 통신 서비스를 효율적으로 제공해 줄 수 있는 새로운 방향을 제시하였다는 것에 만족하고 이 논문을 끝맺는다.

참고 문헌

[1] J. W. Hong, T. H. Yun, J. Y. Kong, and Y. M. Shin, "A Flexible and Reliable Distributed Multimedia System for Multimedia Information Superhighways", *Malaysian Journal of Computer Science*, December 1997.

[2] T. H. Yun, J. Y. Kong and J. W. Hong, "A CORBA-based Distributed Multimedia System", *Proc. of 1997 Pacific Workshop on Distributed Multimedia Systems*, Vancouver, Canada, July, 1997, pp. 1-8.

[3] T. H. Yun, J. Y. Kong and J. W. Hong, "Object-oriented Modeling of Distributed Multimedia Services", *Proc. of IEEE International Conference on Communications*, Montreal, Canada, June, 1997, pp. 777-781.

[4] T. H. Yun, J. Y. Kong and J. W. Hong, "MAESTRO: A CORBA-based Distributed Multimedia System", *Technical Report PIRL-TR-97-2*, POSTECH, March 1997.

[5] J. Y. Kong, J. W. Hong, J. T. Park and D. J. Kim, "A CORBA-Based Management Framework for Distributed Multimedia Services and Applications", *Proc. of the Distributed Systems: Operations and Management*, Sydney, Australia, October 1997, pp. 132-144.

[6] J. Y. Kong, T. H. Yun, S. W. Park, S. H. Kim, Y. M. Shin and J. W. Hong, "Design and Implementation of a Management System for CORBA-Based Distributed Systems", *Proc. of the Asian-Pacific Network Operations and Management Symposium*, Seoul, Korea, October 1997, pp. 473-485.

[7] Ji-Young Kong and J. Won-Ki Hong, "A CORBA-based Management Framework for Distributed Multimedia Services and Applications", *Technical Report PIRL-TR-97-1*, POSTECH, March 1997.

[8] Barry K. Aldred, "IBM Lakes Architecture: Introduction and Programmers' Guide", IBM UK Laboratories Ltd., 1993, URL http://www.hursley.ibm.com/~p2p/.

[9] Interactive Multimedia Association, "IMA Recommended Practice Draft", September 1994, URL http://www.ima.org/forums/imf/mss/.

[10] P. Dubois, "Detailed Specification of the BETEUS application platform", BETEUS Consortium, November 1994. URL http://www.tik.ee.ethz.ch/~beteus/.

[11] Electronics and Telecommunications Research Institute (ETRI), *Mux Tutorial*, 1995.

[12] M. Arango, et al, "The Touring Machine System", *Communications of the ACM*, 36(1):68-77, January 1993.

[13] OMG, The Common Object Request Broker: Architecture and Specification Revision 2.0, OMG, July 1995.

[14] OMG, CORBAservices: Common Object Services Specification, OMG Document Number 95-3-31, March 1995.

[15] OMG, CORBAservices: EventService Specification, http://www.omg.org/corba/sectrans.htm, March 1995.

[16] Microsoft, "DCOM: A Business Overview", http://www.microsoft.com/ntserver/info/dcom.htm, August, 1997.

[17] S. E. Deering, "Multicast Routing in a Datagram Internetwork", (Ph. D. dissertation, Stanford University, Palo Alto, California, Dec. 1991).

[18] S. E. Deering and D. R. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs", *ACM Transactions on Computer Systems*, Vol. 8, No. 2, May. 1990, pp. 85-110.

[19] S. E. Deering, "RFC-1112: Host Extension for IP Multicasting", Aug. 1989.

[20] H. Eriksson, "MBONE: The Multicast Backbone", *Communications of the ACM*, Vol. 37, No. 8, Aug. 1994, pp. 54-60.

[21] J. Postel, "RFC-768: User Datagram Protocol", Aug. 1980.

[22] B. Whetten, T. Montgomery and S. Kaplan, "A High Performance Totally Ordered Multicast Protocol", Aug. 1994, URL ftp://research.ivv.nasa.gov/pub/doc/RMP/RMP_dagstuhl.ps.

[23] S. Floyd, V. Jacobson, S. McCanne, C-G. Liu and L. Zhang, "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing", *Proceedings of ACM SIGCOMM '95*, Aug. 1995, pp. 342-356.

[24] H. W. Holbrook, S. K. Singhal and D. R. Cheriton, "Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation", *Proceedings of ACM SIGCOMM '95*, Aug. 1995, pp. 328-341.

[25] S. Paul, K. K. Sabnani, J. C. Lin and S. Bhattacharyya, "Reliable Multicast Transport Protocol (RMTP)", *IEEE Journal on Selected Areas in Communications, special issue on Network Support for Multipoint Communication*, Vol. 15, No. 3, April 1997, pp. 407-421.

[26] Audio-Video Transport Working Group, "RFC-1889: RTP: A Transport Protocol for Real-Time Applications", Jan. 1996.

[27] OMG, "Control and Management of A/V Streams Request For Proposal", August 1996.

[28] IONA, *Orbix 2*, IONA Technologies Ltd., Release 2.0.

[29] IONA, *OrbixWeb 2.0.1*, IONA Technologies Ltd., Release 2.0.1.

[30] Sun Microsystems, *Sun Video User's Guide*, August 1994.

[31] Sun Microsystems, *Solaris XIL 1.1 Imaging Library Programmer's Guide*, November 1993.

[32] J. Won-Ki Hong. "Distributed Systems Management Technology", *KISS Review*, 14(1):51-61, January 1996.

[33] S. J.Gibbs and D. C. Tsichritzis, *Multimedia Programming, Objects, Environments and Frameworks*, Addison-Wesley, 1995.

[34] E. R. Harold, *Java Network Programming*, O'Reilly & Associates, 1997.

[35] Sun Microsystems, "The Java Virtual Machine Specification", URL http://java.sun.com/docs/books/vmspec/index.html.

[36] Xing Technology Corp., Press Release: StreamWorks 3.0, URL http://www.xingtech.com/about_xing/press_room/072297swrel.html.

[37] RealNetworks Inc., About RealNetworks, URL http://www.real.com/corporate/index.html.

Appendix A. MAESTRO 통신 서비스를 위한 CORBA IDL 정의

```
module Communication {
   typedef sequence<octet> Media;

   struct QOS_Parameter {
       unsigned short bandwidth;
    };

   exception AlreadyUsed { };

   exception AlreadyJoined { };
    exception NotJoined { };

   exception NoParentPort { };
   exception NoParentConnector { };

   exception NoMedia { };

   exception PortNotPrepared { };
   exception DomainNotFound { };
   exception NegotiationFailed { };

   exception CanNotProceed { string reason; };

    interface Connector;
    interface Channel;

   enum CommunicationKind { one_to_many, many_to_many };

   enum PortState { not_joined, joined };

   enum JoinKind { channel_created, connected };

   enum ConnectionKind { intra, inter };

    interface Port {
       readonly attribute string domain_name;

       readonly attribute CommunicationKind communication_kind;

       readonly attribute PortState state;

       readonly attribute JoinKind join_kind;

       readonly attribute ConnectionKind connection_kind;

        /* */
       unsigned short port_number();
      void set_port_number(in unsigned short port_number)
            raises (AlreadyUsed);

        /* */
      void connect(in string domain_name, in unsigned short port_number,
```

```
            in QOS_Parameter q)
          raises (AlreadyJoined, DomainNotFound, PortNotPrepared,
                NegotiationFailed);
      void disconnect();

     void set_parent_port(in Port p);
     void set_parent_connector(in Connector c);
      Port parent_port()
           raises (NoParentPort);
      Connector parent_connector()
           raises (NoParentConnector);

     void add_child_port(in Port p)
           raises (CanNotProceed);
     void remove_child_port(in Port p);

     void add_child_connector(in Connector c)
           raises (CanNotProceed);
     void remove_child_connector(in Connector c);

     void set_channel(in Channel ch);

      /* */
     void send_media(in Media m)
           raises (NotJoined);
      Media return_media()
           raises (NotJoined, NoMedia);

     void route_media(in Media m, in Object from_o);

      /* */
      Channel create_channel()
          raises (AlreadyJoined, CanNotProceed);
      void destroy_channel();

      Channel return_channel()
           raises (NotJoined);
 };

exception InvalidIndex { };

 interface Channel {
     unsigned short port_count();
    Port return_port(in unsigned short index)
           raises (InvalidIndex);

     void set_root_port(in Port p);
      Port root_port();

      void set_real_channel(in Channel ch);

     void join(in Port p)
           raises (CanNotProceed);
     void leave(in Port p);
 };

exception PortNotFound { };
```

```
enum ChannelKind { real, proxy };

interface CommunicationFactory {
    /* */
    Port create_port(in CommunicationKind ck)
        raises (CanNotProceed);
    void destroy_port(in Port p);

    Port search_port(in unsigned short port_number)
        raises (PortNotFound);

    /* */
    Channel create_channel(in ChannelKind ck)
        raises (CanNotProceed);
    void destroy_channel(in Channel ch);

    /* */
    Connector create_connector(in CommunicationKind ck)
        raises (CanNotProceed);
    void destroy_connector(in Connector c);
};

interface Connector {
    readonly attribute string domain_name;

    void set_parent_port(in Port p);

    void add_child_port(in Port p)
        raises (CanNotProceed);
    void remove_child_port(in Port p);

    void route_media(in Media m, in Port from_p);

    Channel return_channel();
    void set_channel(in Channel ch);
};

interface ConnectionManager {
    void connect(in string domain_name, in unsigned short port_number,
        in Port child, in QOS_Parameter q)
        raises (DomainNotFound, PortNotPrepared, NegotiationFailed, CanNotProceed);
    void disconnect(in Port p);

    boolean port_preparation(in unsigned short port_number, in CommunicationKind ck);
    void negotiate(in QOS_Parameter q)
        raises (NegotiationFailed);
    void inter_connect(in Connector c);
};
};
```

감사의 글

2 년 동안 분산 처리 연구실 가족들과 소중한 시간을 보내었습니다. 이제 졸업을 앞두고 분산 처리 연구실을 떠나게 되어 아쉬움이 앞섭니다. 분산 처리 연구실의 여러 선배들과 동기들 그리고 후배들과 함께 보낸 시간들은 저에게 너무나 소중한 기억들입니다. 제가 앞으로 다른 곳에서 일하게 되더라도 분산 처리 연구실 가족들과 함께 한 시간을 잊지 못할 것입니다.

그리고 2 년 동안 부족한 저를 신임해 주시고 믿고 일을 맡겨 주신 홍원기 교수님께 깊은 감사를 드립니다. 돌이켜 보면 2 년 전의 저는 참 부족한 점이 많았습니다. 그럼에도 불구하고 저를 지도하고 이끌어 주시느라 귀중한 시간을 내어 주신 교수님께 정말 다시 한 번 깊은 감사를 드립니다.

그리고 바쁘신 시간을 쪼개어 논문 심사를 맡아 주신 강교철 교수님과 김정현 교수님께도 감사를 드립니다.

앞으로 분산 처리 연구실의 무궁한 발전을 기원하며 감사의 글을 마칩니다.

이력서

성명 : 윤태형

생년월일 : 1973 년 10 월 12 일

출생지 : 부산시 부암동 333 번지

주소 : 경북 포항시 남구 지곡동 756 번지 포항공대 기숙사 18 동 101 호

학력

1992 – 1996 : 포항공과대학교 전자계산학과 (학사)

1996 – 1998 : 포항공과대학교 전자계산학과 (석사)

학술 활동

Journal Papers

✍✍J. W. Hong, T. H. Yun, J. Y. Kong, and Y. M. Shin, "A Flexible and Reliable Distributed Multimedia System for Multimedia Information Superhighways", *Malaysian Journal of Computer Science*, December 1997.

✍✍J. W. Hong, J. Y. Kong, T. H. Yun, J. S. Kim, J. T. Park and J. W. Baek, "Web-based Intranet Services and Network Management", *IEEE Communications Magazine*, October 1997, pp. 100-110 (SCI).

Conference Papers

✍✍T. H. Yun, J. Y. Kong and J. W. Hong, "A CORBA-based Distributed Multimedia System", *Proc. of the Fourth Pacific Workshop on Distributed Multimedia Systems*, Vancouver, Canada, July 1997, pp. 1-8.

✍✍T. H. Yun, J. Y. Kong and J. W. Hong, "Object-Oriented Modeling of Distributed Multimedia Services", *Proc. of the the IEEE International Conference on Communications (ICC)*, Montreal, Canada, June 1997, pp. 777-781.

✍✍윤태형, 공지영, 홍원기, "분산 멀티미디어 서비스의 모델링 (Modeling of Distributed Multimedia Services)", *Proc. of the 23rd KISS Fall Conference*, Youngin, Korea, October 1996, pp. 1313-1316.

✍✍윤태형, 홍원기, "WWW 에서 포항공대 전자신문의 디자인과 구현", *The 3rd WWW Korea Workshop*, Pusan, Korea, March 22-23, 1996, pp. 243-252.

Technical Reports

✍✍T. H. Yun, J. Y. Kong and J. W. Hong, "MAESTRO: A CORBA-based Distributed Multimedia System", *POSTECH PIRL Technical Report*, PIRL-TR-97-2, March 1997.