CORBA 환경에서 JAVA 를 이용한 공동 작업 환경 구현

○
김 근형, 이 봉환*
한국통신 네트워크 연구소 초고속 통신 연구팀
*대전대학교 정보 통신학과

Implementation of Shared Work Environment using JAVA on the CORBA Environment

○
Geunhyung Kim, Bong-Hwan Lee*
Broadband Network Engineering Division, Telecommunication Network Laboratory, Korea Telecom
* Dept. of Information and Communication, Taejon University

요 약

멀티미디어 회의, 원격 교육, 원격 의료 등과 같이 다 수가 참여하여 공동으로 작업을 하는 응용 서비스들이 컴퓨터와 네트워크의 발전에 따라 가능해지고 있다. 이와 같이 다수가 참여하여 객체를 공유하면서 공동 작업을 하여야 하는 응용들은 공유 작업 환경(shared work environment)을 필수적으로 요구한다. 또한 네트워크에서 다양한 응용 서비스들을 제공하고 관리하기 위해서는 서비스 세션에 대한 관리도 필요하다. 본 논문에서는 사용자가 여러 세션에 참여하여 공동 작업이 가능한 공유 작업 환경을 다자간 멀티미디어 서비스 세션의 제공, 제어 및 관리를 위한 메커니즘을 제공하는 CORBA(Common Object Request Broker Architecture)기반의 COMUS(CORBA based Multiparty Multimedia System)와 자바 언어의 객체 지향 특성을 이용하여 구현하였다.

1. 서론

통신 기술의 발전과 네트워크의 고속화에 따라, 지리적으로 떨어진 사용자들이 통신 망을 통해 공동 작업을 하는 것이 가능하게 되었다. 여러 사용자들이 공동 작업을 진행하기 위해서 사용되는 서비스들로는 멀티미디어 메일, 데이터 회의, 영상회의, 원격 강의와 같은 것들이 있다. 이와 같은 공동 작업을 위해서 필수적으로 요구되는 기반 구조로는 통신 망에 분산된 사용자들이 공동의 멀티미디어 객체들에 대해 메소드를 요구할 수 있는 공유 작업 환경이다. 이 공유 작업 환경을 이용하여 참여자들은 문제를 분석하고, 해결하기 위해 서로의 아이디어와 정보를 교환하게 된다. 이와 같은 공유 작업 환경을 효과적으로 제공하기 위해서 네트워크를 통해 분산된 사용자 들간의 통신을 효과적으로 가능하도록 하는 분산 플랫폼의 구축이 필요하다.

분산 플랫폼들은 이기종 시스템 및 응용들을 쉽게 통합하고 일관된 방법으로 개발을 할 수 있도록 보장하는 다양한 미들웨어들로 구성된다. 여러 미들웨어들 중 객체 지향 패러다임을 통해 분산 통합 환경을 제공하여 주는 OMG CORBA 가 여러 이유에서 기존의 어떤 미들웨어보다 우수하다

[1]. 최근에는 CORBA를 이용한 멀티미디어 응용 서비스의 개발 및 멀티미디어 통신 플랫폼에 대한 연구도 진행 중이다[2,3].

기존의 공유 작업 환경은 같은 하드웨어 플랫폼 간에서만 제공되었으며, 통신 망 내의 서로 다른 하드웨어 플랫폼을 사용하는 참여자들이 작업 공간을 공유하는 것은 어려웠다. 그러나 최근 하드웨어 플랫폼에 무관한 바이트 코드를 생성시키는 자바 언어로 인해 자바 가상 머신(Java Virtual Machine)이 구현된 모든 하드웨어 플랫폼에서 자바 언어를 사용한 응용 프로그램의 수행이 가능하게 되었다.

또한 네트워크이 광대역화 됨에 따라 기존에는 생각지도 못했던 다양한 멀티미디어 응용 서비스들이 네트워크 상에서 존재하고 있다. 이와 같이 다양한 멀티미디어 서비스들을 효과적으로 제공하고 관리하기 위해서는 세션의 관리 기능을 필요로 한다.

본 논문에서는 CSCW(Computer Supported Cooperative Working), 멀티미디어 회의, 원격 강의 등과 같은 다자간 멀티미디어 서비스들의 서비스 세션을 관리하는 기반으로 [3]에서 제안한 COMUS 상에서 공유 작업 환경을 CORBA 환경에서 자바 언어를 사용하여 구현하였다.

본 논문의 구성은 다음과 같다. 2 장에서는 본 논문에서 제시한 공유 작업 환경과 관련된 연구들에 대하여 살펴보고, 3 장에서는 2 장에서 설명한 기술들을 사용하여 설계한 공유 작업 환경에 대하여 기술한다. 4 장에서 실제 공유 작업 환경을 이용한 공유 작업 공간의 구현 예를 살펴보고 결론을 맺는다

2. 관련 연구

2.1 CORBA

CORBA는 객체 지향 기술을 바탕으로, 응용 프로그램들이 통신 망에 어디에 위치하든지, 어떤 프로그래밍 언어로 구현되든 관계없이 다양한 하드웨어 플랫폼과 운영 체제를 지원함으로써 오늘날 분산 컴퓨팅 환경을 지원하는 새로운 시스템 통합 기술이다.

CORBA 구조[1]는 그림 1 과 같고 구성 요소들은 다음과 같은 기능들을 갖는다.

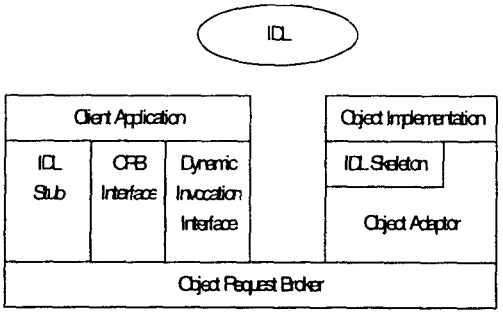

그림 1. CORBA 구조

IDL(Interface Definition Language) : 객체의 인터페이스를 정의하기 위한 언어로, 기본 데이터 형식, 구조화된 데이터 형식, 템플릿 데이터 형식을 제공한다.

ORB: 요구 및 응답 메시지를 네트워크를 통해 분산되어 있는 해당 서버 및 클라이언트에게 정확히 전달해 주는 통신 메커니즘을 제공한다.

Object 어댑터: 서버의 구현 객체를 인스턴스화 할 수 있는 런-타임 환경을 제공하고, 서버의 구현 객체로 메시지를 전달하며, 객체 식별자를 부여하는 역할을 한다.

DII: 클라이언트가 런 타임 시 메소드를 동적으로 찾을 수 있도록 하는 기능을 제공한다.

ORB Interface: 응용에서 필요할 수도 있는 로컬 서비스들이 정의되어 있다.(예: object_to_string() 또는 string_to_object())

2.2 JAVA

자바 언어는 간단하고 단순한 객체 지향 언어로서 아키텍처에 독립적이고, 이식성이 높으며, 인터프리터 방식의 동적인 언어이다[5].

자바 언어는 가상 머신 개념에 기반을 두고 있는 언어로 자바 컴파일러에 의해서 생성된 바이트 코드가 자바 해석기에 의해서 해석된다. 자바의 바

이트 코드는 플랫폼에 독립적이기 때문에 자바 해석기가 있는 하드웨어 플랫폼에는 모든 이식이 가능하다. 그리고 자바 언어는 매우 단순하기 때문에 기본적인 해석기의 크기가 40 Kbytes 정도이면 된다(기본적인 표준 라이브러리를 포함하기 위해서는 175 Kbytes 정도가 더 필요하다.) 이와 같은 자바의 특성 때문에 자바 언어를 이용하여 네트워크 PC 또는 VoD 용 STU 와 같이 제한된 자원을 갖는 장치들에서 실행되는 응용 프로그램들의 사용자 인터페이스로 사용되고 있다.

2.3 COMUS

COMUS 는 CORBA 환경에서 다자간 멀티미디어 통신이 가능하도록 제시한 다자간 멀티미디어 통신 플랫폼[3]이다.

플랫폼의 아키텍처는 그림 2 와 같이 계층적 아키텍처로서 개방성을 보장할 수 있도록 정의하였다. 아키텍처의 최하위 계층에는 실제 프로세서와 운영체제와 같은 컴퓨팅 자원들과 실제 분산 멀티미디어 응용에 참여하는 사용자들을 연결하는 네트워크 기술(ATM, Ethernet, Token Ring 등)들이 위치한다.

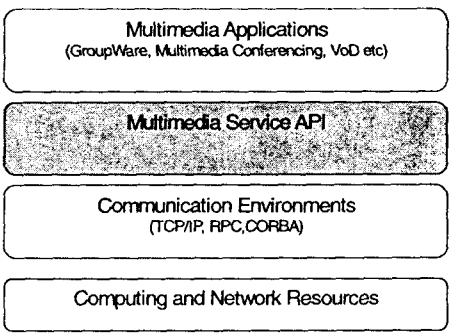

그림 2. 멀티미디어 통신 플랫폼 아키텍처

다음 두 번째 계층은 분산 통신 환경을 제공하는 계층으로 이와 같은 환경을 제공하는 것은 OSF(Open Software Foundation)의 DCE(Distributed Computing Environment), RPC(Remote Procedure Call), CORBA 의 ORB(Object Request Broker)와 같은 것들이 있다. 세 번째 계층은 두 번째 계층의 통신 환경들을 사용하여 멀티미디어 데이터를 전달하고 세션을 생성/제어하는 기반 서비스를 제공하는 계층이다. COMUS 는 세 번째 계층에서 정의된 서비스를 제공하는 플랫폼으로 객체 지향 기법을 사용하여 설계하였으며, 분산된 멀티미디어 응용들이 이 계층에서 제공하는 API 를 통하여 하위 계층에 투명하게 상호 연결 및 데이터 전달을 할 수 있다. 최상위 계층은 그룹웨어, 멀티미디어 회의, whiteboard 등과 같은 다양한 멀티미디어 응용들에 해당된다.

본 COMUS 는 멀티 서비스들의 세션을 관리하기 위해서 크게 서비스 게이트웨이, 세션 관리 서버, 클라이언트 세션 관리자, 세션 어댑터로 구성된다. 서비스 게이트웨이는 네트워크상에서 등록된 서비스들에 대한 정보를 제공하고, 각 서비스 마다 세션 관리 서버가 존재하여 서비스 별 세션을 관리하고 있다. 또한 클라이언트에는 세션 어댑터가 존재하여 응용 프로그램 개발자에게 세션과 관련한 Java API 를 제공한다. 세션 어댑터가 제공하는 Java API 들은 다음과 같다.

void GetSessionsList(SessionServer sm, SessionDescListHolder sessionInfo);

void CreateSession(SessionServer sm, String owner, String serviceid, String videotype, String audiotype, String whiteboard, ComusSessionType sessionType, String uuData, SessionIdHolder sessionid);

boolean JoinSession(SessionServer sm, String sessionid, String userid, String tapid, String napid, String videotype, String audiotype, String whiteboard, String uuData);

boolean ReleaseSession(SessionServer sm, String sessionid, String userid, String why, String uuData);

3. 공유 작업 환경의 설계

제시한 공유 작업 환경은 CORBA 환경에서 여러 사용자들이 하나의 객체를 대상으로 공동 작업을 할 수 있도록 하는 기반을 제공한다. 본 논문의 공유 작업 환경의 구조는 다음 그림 3 과 같다.

그림 3 의 공유 작업 환경에서는 네트워크 상에 분산된 사용자들에게 공유 작업을 지원하기 위해 COMUS 에서 정의한 세션 제어 메커니즘을 사용하여 각 사용자들이 원하는 공유 작업 세션에 참여/

탈퇴 및 세션의 일시 정지 및 재개 기능을 제공한다.

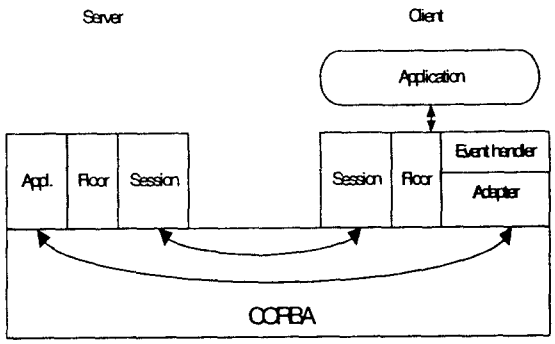

그림 3. 공유 작업 환경 구조

또한 각 참여자로부터 전달되어 오는 이벤트들을 세션 내의 다른 참여자들에게 전달해주는 구조로는 복제 구조(replicated architecture)를 사용한다. 이 구조는 네트워크를 통해 이벤트 만을 전달하기 때문에 네트워크 상에 트래픽이 적고 뛰어난 성능을 보인다. 이밖에 공유 작업 환경에서 필요한 것이 floor 제어이다. 본 공유 작업 환경에서는 floor를 요청하면 즉시 제공하는 것으로 단순화 하였다. 공유 작업 전반을 제어/관리 하기 위해서 서버 측에는 세션 객체, 응용 객체, floor 객체들로 이루어진다.

또한 사용자의 작업 내용이 모든 사용자들의 공유 작업 공간에 나타나도록 하기 위해서 사용자의 요구에 따라 발생하는 이벤트들을 같은 세션에 참여하는 모든 참여자들의 공동 작업 공간에 똑같이 나타나는 구조를 갖추기 위해서 그림 4와 같은 이벤트 공유 절차를 따른다.

공유 절차에서 이벤트 처리기(Event handler)가 하는 역할은 응용으로부터 전달된 이벤트를 가로채서 로컬 이벤트를 생성시키고, 어댑터를 통해 네트워크 상의 이벤트 버스인 commBus에게 가로챈 이벤트를 전달한다. commBus는 같은 세션에 참여한 참여자의 어댑터에게 이벤트를 전달한다. 다른 참여자(즉 floor를 갖지 않은 참여자)들의 이벤트 처리기는 네트워크로부터 전달된 이벤트에 따라 내부 이벤트를 발생시킨다. 이와 같이 이벤트를 가로채는 메커니즘은 객체 지향 프로그래밍 언어의 특성인 상속 기능을 통해, 프로그래밍 언어의 이벤트 처리기를 상속 받아 구현한다.

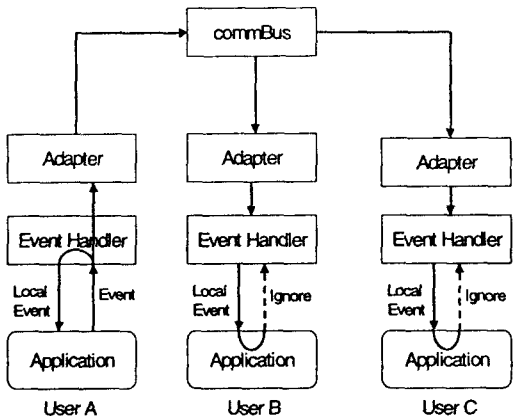

그림 4. 이벤트 공유 절차

자바 또는 C++와 같은 객체 지향 프로그래밍 언어들을 객체의 상속을 통해 객체의 정의를 재사용할 수 있다는 장점을 가진다. 자바의 AWT(Abstract Windowing Toolkit) 또는 MFC(Microsoft Foundation Classes)들과 같은 클래스 군이 제공하는 이벤트 객체들을 상속 받아서 그림 4의 이벤트 처리기를 정의하면, 이벤트 처리기는 쉽게 응용 프로그램에서 발생한 이벤트들을 전달받을 수 있다. 응용에서 발생한 이벤트를 전달 받은 이벤트 처리기는 어댑터를 통해 네트워크에 존재하는 이벤트 버스에 이벤트를 전달하고, 이벤트에 대한 처리를 수행하도록 한다.

또한 이벤트 버스 역할을 하는 commBus는 다음과 같은 IDL 인터페이스로 정의된다.

```
interface commBus{
 oneway void SendMessage(in SessionId sid, in String event);
 oneway void RegisterClient(in SessionId sid, in Object objrefs);
 oneway void RemoveClient(in SessionId sid, in UserId uid);
};
```

4. 공유 작업 환경 구현

공유 작업 환경의 구현은 그림 5와 같다. 여기서 세션 관리자, 흐름제어 및 응용 서버, 서비스 게이트웨이로 사용하는 하드웨어 플랫폼은 SUN사의 Sparc 20 워크스테이션(운영체제: Solaris 2.51, C/C++ 4.0 이상, ORB로는 Iona사의 OrbixMT 2.2.1)이며, 사용자들은 Iona사의 OrbixWeb3.0과 JDK 1.1.6 이상이 설치된 개인용 컴퓨터나 워크스테이션을 통해 공유 작업이 가능하게 된다.

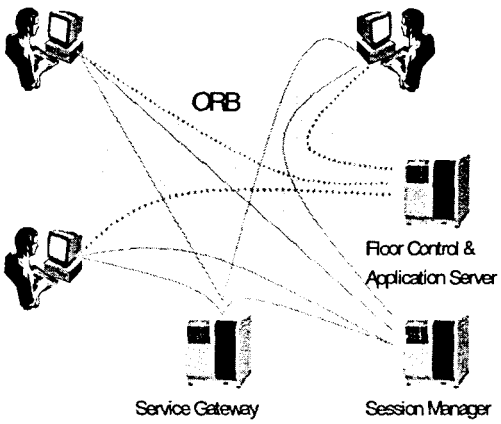

그림 5. 공유 작업 환경 구현 시스템 구성 예

먼저 각 사용자들은 서비스 게이트웨이를 통해 네트워크 상에 등록된 서비스들의 객체 레퍼런스를 얻게 되고 자신에 대한 객체 레퍼런스도 서비스 게이트웨어를 통해 해당 서비스 context에 등록하게 된다. 다음 그림 6에는 사용자와 서비스 게이트웨어와의 상호 작용을 나타냈다.

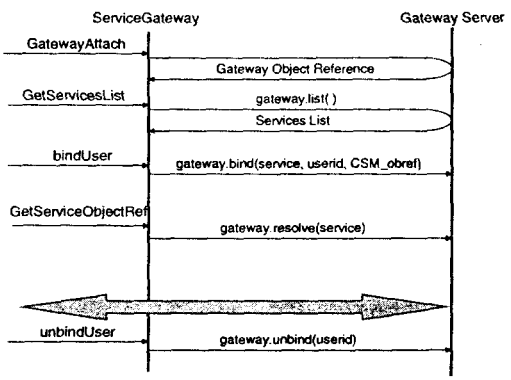

그림 6. 서비스 게이트웨이와의 상호 작용

그림 6에서 서비스 게이트웨이는 클라이언트에 존재하는 게이트웨이 서버의 프록시 객체로서 상위 응용 계층으로부터 전달된 메소드들을 실제 네트워크를 통해 원격지의 객체 서버와 통신을 하게 된다. 서비스 게이트웨이 객체가 제공하는 Java API들은 다음과 같다.

void GatewayAttach(String gatewayserver);

void GetServicesList(int size, BindingListHolder list, BindingIteratorHolder Iter);

SessionServer GetServiceObjectRef(String selectedService);

void bindUser(String selectedService, String userId, org.omg.CORBA.Object objRef);

void GetUsersList (String selectedService, BindingListHolder list);

void unbindUser(String selectedService, String userId);

사용자는 서비스 게이트웨이를 통해 자신이 원하는 서비스를 선택하고 그 서비스의 세션 관리자를 통해 현재 생성된 세션 정보를 얻게 되며 또한 새로운 세션을 생성하거나 기존의 세션에 참여하게 된다. 세션과 관련된 Java API들은 2.3 절에 살펴보았으며, 세션 계층의 서비스를 제공하는 객체들의 관계를 그림 7에 나타냈다.

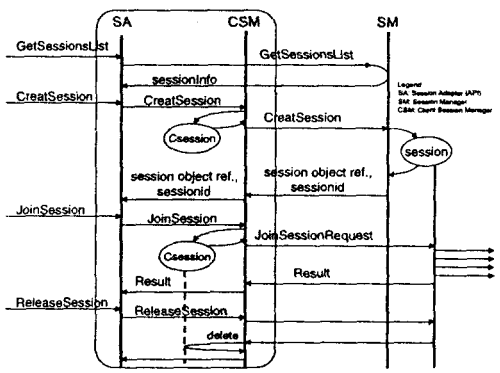

그림 7. 세션 계층 구성 요소 및 프리미티브들

그림 7의 SM은 세션 관리자로서 세션에 관한 정보들을 유지하고 관리하는 역할을 담당한다. SM은 여러 세션을 관리하기 위하여 세션 객체의 팩토리 객체 역할을 하며, 각 세션 객체들이 실제 세션에 대한 전반적인 정보를 유지한다. 또한 CSM은 클라이언트의 세션 관리자로 사용자에 대한 에

이전시 역할을 담당한다. 또한 SA는 세션 어댑터로서 클라이언트의 응용 개발자들에게 세션에 대한 API를 제공하는 역할을 한다.

앞에서 언급한 세션 계층의 프리미티브들과 서비스 게이트웨이들을 이용하여 간략히 구현한 클라이언트의 화면 중 하나를 그림 8에 나타냈다. 그림 8은 현재 세션에 대한 상세 정보를 보여 주는 화면으로 현재 세션의 소유자, 세션 타입, 참여자, 세션에서 사용중인 응용 프로그램들을 표시한다. 이 정보를 기반으로 진행중인 세션에 참여할 수 있다.

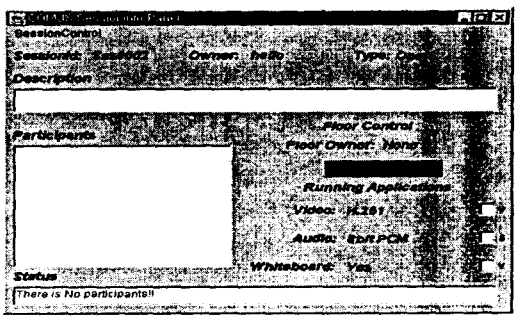

그림 8. 한 세션에 대한 상세 정보 화면

그림 7에는 COMUS 상에서 자바로 작성한 응용 프로그램 중의 하나인 공유 작업 공간의 예를 보였다.

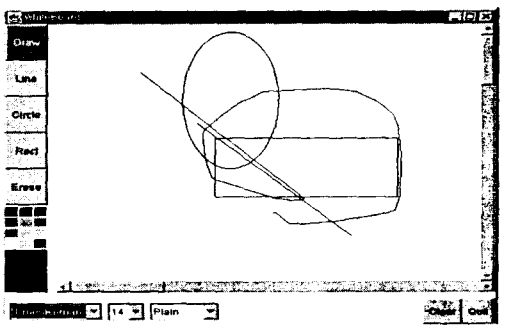

그림 7. 공유 작업 환경을 이용한 공유 작업 공간

5. 결론

본 논문에서는 CORBA 환경에서 멀티미디어 회의, 데이터 회의, 원격 강의와 같은 공동 작업을 위해 사용되는 응용 프로그램들에게 공유 작업 환경을 제시하였다.

제시한 공유 작업 환경은 공유 작업 공간과 같은 공유 작업을 위한 환경은 CORBA.환경에서 설계한 다자간 멀티미디어 통신 플랫폼인 COMUS 상에서 구현 된 것으로 공유 작업 세션에 대한 제어를 하기 때문에 사용자가 여러 공동 작업 세션에 동시에 참여할 수 있도록 하였다.

향 후에는 본 논문에서 제시한 이벤트 버스와 멀티캐스트 기능과의 결합에 대하여 살펴보고, CORBA 이벤트 서비스를 공유 작업 환경의 이벤트 버스로 적용해볼 예정이다. 또한 공동 작업을 위한 floor 제어 메커니즘에 대한 연구도 진행할 예정이다. 또한 자바 언어의 특성을 고려하여 클라이언트 응용 프로그램을 서버로부터 다운로드할 수 있는 방법을 고려할 것이다.

참고문헌

[1] Robert Orfali, *Instant CORBA*, 1st edition, Wiley, 1997.

[2] 김 근형 외 2인, "CORBA 환경에서의 대화형 분산 비디오 검색 시스템 구현", 한국통신학회 1997년도 추계 종합학술발표회 논문집, 1997년 11월.

[3] 김 근형 외 4인,"CORBA 환경에서의 다자간 멀티미디어 통신 플랫폼 설계" 한국통신학회 1998년 하계 종합학술 발표회 논문집, 1998년.

[4] Jon Siegel, *CORBA Fundamentals and Programming*, 1st edition, Wiley, 1996.

[5] David Flanagan, *Java in a nutshell*, 2nd edition, O'Reilly, 1997.