CORBA의 분산된 객체구현들간의 직접적인 데이타 전송

한정규*, 채수환
한국항공대학교 전자계산학과

Direct Data Transfer Between Distributed Object Implementation in CORBA

Jung Gyu Han*, Soo Whan Chae
Dept.of Computer Science, Hankuk Aviation University

요 약

분산 멀티미디어시스템 응용 소프트웨어를 개발함에 있어 그 효율적인 동작을 목적으로 하는 다양한 연구가 활발히 진행되고있다. 이는 곧 클라이언트-서버 환경을 어떻게 구성 할 것인가의 문제를 의미한다. 이러한 문제를 해결하기 위한 방법론을 제공해줄 수 있는 강력한 시스템 하부 기반기술을 찾는것이 응용프로그래머들의 당면 문제 였다. 이러한 차원에서 다양한 시스템 소프트웨어가 개발 되었지만, 우리는 객체 지향 개발에 가장 적합한 CORBA를 선택하게 되었다. CORBA는 프로그래머에게 네트워크 하부 구조에 대한 확고한 투명성을 제공함으로써 클라이언트-서버 구조를 구성하는데 편의성을 제공한다. 그러나 다단계의 클라이언트-서버 구조를 이용한 기존 응용들에서는 서버들간의 데이타 전송에 클라이언트가 관여함으로써 전송상의 불가피한 오버헤드를 가지고 있었다. 이러한 문제를 서버들간의 직접적인 데이타 전송으로 해결하고자 한다. 본 논문에서는 객체화 되어 있는 CORBA 구현들간의 직접전송 방법론을 제시하고 그 실제 개발 예를 소개하고자 한다.

1. 서 론

초고속 정보통신망용 응용소프트웨어들이 공통적으로 직면하고 있는 가장 어려운 문제점은 새로운 응용 프로그램을 위한 개발도구와 하부 시스템 소프트웨어가 부족하다는 점이다. 그중에서도 다양한 멀티미디어 데이타들을 이용한 응용소프트웨어들의 경우는 더욱 그러하다. 기존의 단일 시스템에서의 멀티미디어 응용 소프트웨어의 경우, 물리적으로 하나의 호스트에 데이타의 생성자(Producer)와 소비자(Consumer)등의 입출력시스템 제어 및 데이타처리 관련기능들이 집중되어 있었다. 이러한 기능들을 독립적인 기능을 갖는 객체(Object)로 분리시키기 위하여 응용프로그래머들은 이미 80년대 중반부터 거론되기 시작했던 객체지향 프로그래밍 기법[1,2]에 다시 관심을 갖기 시작했다. 이중에서도 객체공장(Object Factory)기법을 도입한 Objective-C[2]와 Smalltalk-80[3]의 객체풀(Object Pool) 개념이 그 대표적인 예이다. 이러한 기법들은 응용프로그램 개발시에 다양한 기능을 가지는 객체들을 하부에서 제공하고 이를 단순히 조합함으로써 새로운 서어비스를 할 수 있는 형태로 발전하게 되었다. 최근 활성화 되고 있는 분산 멀티미디어 시스템의 경우, 지리적으로 분산되어 있는 리소스들을 어떻게 하면 효율적으로 관리하고 균형있게 사용 할 수 있는가 하는 것이 더욱 큰 관심사로 등장하게 되었고, 결국 다양한 주소공간을 갖는 시스템들을 하나의 시스템처럼 취급할 수 있도록해주는 방법론이 필요하게 되었다. 이러한 문제점들을 종합적으로 해결하기 위한 많은 연구가 90년대 초반부터 진행되어 왔다. 그중 대표적인 것이 Microsoft사의 COM(Common Object Manager), IBM사의 SOM(System Object Manager), OMG(Object Mangement Group)의 CORBA(Common Object Request Broker Architecture)이다. 이들은 모두 응용프로그래머

가 네트워크 프레임워크(Network FrameWork)자체 보다는 분산 객체 관리를 위한 고급 프로그래밍 에 촛점을 맞출수 있도록 투명성(Transparancy)을 제공한다. 이중 SOM과 CORBA는 동적 인터페이스 호출(DII), Smart Binding, Smart Protocol Selection 등의 유연성 있는 기능을 제공한다. 본 논문에서는 CORBA 사양의 완벽한 구현 형태인 Orbeline[5]을 이용하여 분산 멀티미디어 시스템에서의 객체구현들간의 직접적인 데이타 전송을 위한 방법론을 제공한다. 또 이를 기반으로 실제 응용의 예를 소개하고, 마지막으로 결론과 앞으로의 개발 전망을 정리하겠다.

2. 객체 구현들간의 직접적인 데이타 전송

현재 우리는 CSCW(Computer Supported Common Working)의 한 형태인 화상회의 시스템(VideoConference System) 프로젝트를 수행하고 있다. 원격지의 시스템간에 원할한 멀티미디어 데이타의 전송을 가능케 함으로써 사용자들이 마치 한 자리에 모여 회의를 하는 듯 한 효과를 주는것이 그 핵심적인 요구이다. 이러한 요구를 만족시키기 위해 CORBA가 지원하는 클라이언트-서버 환경을 이용하였다. 일반적인 클라이언트-서버 환경에서는 멀티미디어데이타 처리를 위한 서버가 존재하여 이 서버내에서 다중 클라이언트를 상대로 스트림, 정보공유, 동기등의 문제를 복합적으로 해결하였다. 이러한 구조에서는 특정 서버에서 처리된 일련의 데이타가 다음처리를 수행하는 어플리케이션의 클라이언트로 복사가 되고 이 데이타가 다시 다른 객체 구현에 할당 될수 있다는 문제점을 갖고 있었다. 이는 불필요한 전송 경로를 포함하게되는 전송상의 오버헤드(Overhead)가 된다. 이러한 문제점을 우리는 클라이언트-객체구현으로 연결된 데이터 경로를 항상 경유하는 시스템이 아닌 객체구현간의 직접 데이타 전송 방식으로 해결하고자 한다. 필요에 따라 객체구현과 객체구현이 직접 멀티미디어 데이터와 같은 방대한 량의 정보를 직접 주고 받도록 하자는 것이다. [그림 1]에서처럼 독립적인 기능을 수행하는 서버가 하나의 어플리케이션으로 구성되고 이들은 시스템에 의존하지 않고 독립적으로 언제, 어느 시스템에서든 수행가능한 상태가 된다. 이는 분산시스템의 Load Balancing 측면에서 매우 중요한 개념이다[4]. 즉, 로컬(Local)에서의 리소스에 대한 사용 부담을 할당된 작업이 없는 원격지의 시스템에 분담시켜 전체 시스템의 효율적인 리소스 관리를 가능케 한다는 것이다[3]. 이러한 서버 어플리케이션들이 모여 개념적 구조인 하나의 서버풀(Server Pool)을 이루고 전체 어플리케이션 시스템의 메인프레임(Mainframe)격인 클라이언트가 존재해 개개의 서버들에 대해 특정 작업을 의뢰 할 수있다. 물론, 이러한 구조를 만들어 내는데는 CORBA의 역할이 중요하다. 하지만 CORBA는 네트워크상의 전송 프로토콜을 자동으로 제공 할 뿐 어플리케이션 자체를 지원하는것이 아니라는것을 밝혀둔다. 이제 전송상의 오버헤드라는 문제점의 해결에는 어느정도 접근하였다.

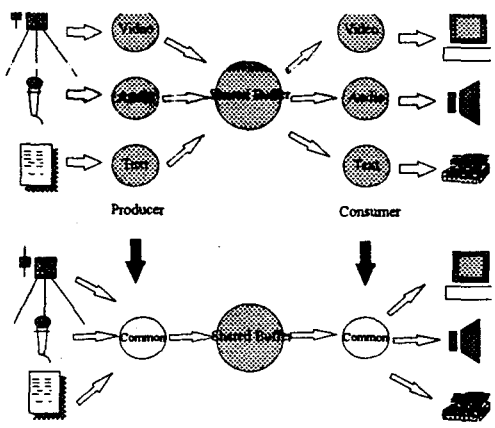
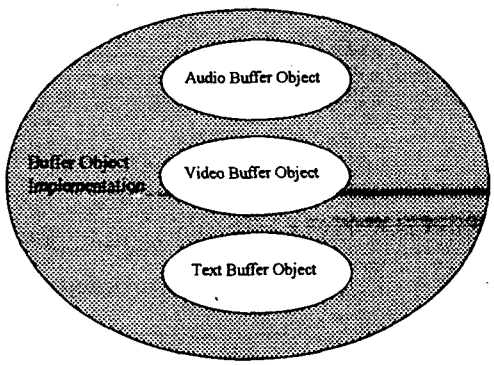


그림 1

하지만 더욱 중요한 문제가 남는다. 어떻게 하면 앞에서 만들어진 서버들에 대해 강력한 확장성(Scalability)과 유연성(Flexibility)을 부여 할것인가의 문제이다. 기존에 방식에서는 유사한 기능을 갖는 객체들이 처리하고자하는 데이타의 타입(Tpye)에 따라 서로 다르게 취급되었다. 오디오(Audio)데이타와 비디오(Video)데이타를 생성하는 객체에 대해 생각해 보자. 만일 하나의 객체가 하나의 어플리케이션을 이뤄야한다면 두개의 객체는 서로 다른 모양의 서버가 될것이다. 이러한 두개의 서버를 하나의 서버로 통합하고자 한다. 여기서 중요한 사실은 이러한 서버의 구체적 모습은 컴파일시(Compile time)에 결정되는 것이 아니라 실행중(Runtime)에 결정되어야한다는 것이다. 즉, 최초의 실행되어 있는 서버는 여러 형태로 변환 될수있는 잠재성(Potential)을 갖는다. 이때 이 서버의 구체적 모습은 메인프레임 클라이언트에 의해 결정되어진다. 이것이 다중 객체구현의 기본개념이다. 앞의 비디오, 오디오 데이타의예로 그려진 그림[2-2]에서는, 객체구현이 기존의 서버구조에 비해 보다 많은 유연성을 갖는다는것을 직관적으로 보여준다.

물론 앞에서 설명된 객체구현이 모든 어플리케이션에 범용으로 수행되는 것은 아니다. 객체구현들을 개발하기에 앞서 어플리케이션의 도메인(Domain)을 제한하는것이 작업이 우선 선행되어야 한다. 우리가 제시한 객체구현 서버는 화상회의라는 어플리케이션 도메인에서 요구되는 멀티미디어 데이타와 그 처리(Processing) 및 데이타 동기화에 필요한 기법들을 정적인 구조가 아닌 동적인 구조로 구현한것에 그 의미가 있다.

이제 이 객체구현서버의 실제 응용의 예를 다음절에서 소개하도록 하겠다.



<그림 2>. 다중화된 객체 구현

3. 분산 멀티미디어시스템에서의 객체구현

다양한 멀티미디어 데이타를 취급함에 있어 필요한 데이타 서버들간의 효율적 전송을 위한 공유 버퍼 서버에 대해 살펴 보도록 하자.

버퍼 서버는 실시간 처리 시스템에 있어 데이타의 생성자 측과 소비자측이 동시에 공유 할 수 있는 방법론을 포함함으로써 두 객체구현 사이의 전송속도차를 해결하기 위한 기본 목적외에도, 중요한 두가지 특성을 갖는다. 첫째 객체구현화 되어있는 버퍼 서버는 시스템에 비의존적으로 동일한 기능을 수행 함으로써 분산시스템의 요구에 적합한 형태를 갖는다. 둘째로 하나의 객체구현 안에 다양한 데이타들에 대한 서어비스 객체들을 포함하고, 실행시간에 동적으로 제공한다[그림 2].

이제 CORBA의 인터페이스 제너레이션을 위한 IDL(Interface Definition Language)을 살펴보자.

<그림 3>의 상부 structure는 전송하고자 하는 데이타의 구조를 명세한다. 또 Buffer는 이 서버가 갖는 공통객체인 인터페이스의 이름이다. 이제 버퍼서버는 Buffer라는 이름으로 다른 서버들과 인터페이스 한다. 이 버퍼서버에 동시에 접근하는 데이타 서버들에 대한 공유문제는 논문의 범주를 넘어서는 프로그래밍 내적인 문제이므로 여기서는 다루지 않겠다. Send_Multimedia_Data는 버퍼서버와 데이타를 생성하는 서버와의 사이에 이용되는 인터페이스 함수이다. 이 함수를 통해 데이타 생성 서버는 실제 데이타와 데이타 타입,데이타 크기등의 부가적인 정보를 버퍼서버로 전송한다. 버퍼서버는 실행시에 데이타의 타입을 동적으로 감지, 해당 데이타에 해당하는 내부적인 오버로딩된 함수를 호출 함으로써 버퍼에 데이타를 저장하고 함수의 실행이 정상적으로 완료되어는지의 여부를 송신측에 알려준다. 데이타의 소비를 맡은 서버에서는 Receive_Multimedia_Data 함수를 이용 버퍼서버에게 버퍼의 값을 전송하도록 메세지를 보내고

```
//IDL for Buffer Implementaion

struct Multimedia_Data {

        short long   Data_Type;
        long   Size;
        ..................

};

Interface Buffer {

boolean Send_M_Data(in M_Data Send_Info);
boolean Receive_M_Data(out M_Data Receive_Info);

};
```

<그림 3> Buffer.IDL

Data_Info 라는 구조형 인스턴스 변수로 전달받는다. 부가적으로 동적인 인터페이스 호출, 베이직 오브젝트 아답터,등 CORBA가 지원하는 기능들을 이용하면 더욱 강력한 형태의 서버 기반 시스템을 구성 할 수 있다.

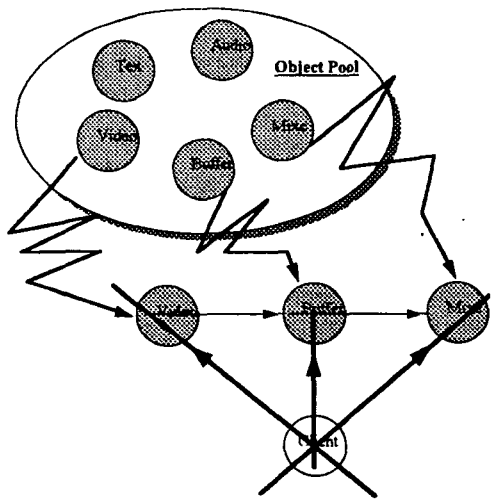


그림 4

이제 이러한 부분적인 요소들을 전체 시스템의 관점에서 살펴 보도록한다. <그림 4>는 객체 구현 서버들을 기반으로 한 분산 멀티미디어 시스템의 개념적인

전체 구도이다. 여기서 Object Pool 은 시스템 내에서의 개념적인 구조이다. 즉 어느한 시스템에 국한되어있는 구조가 아니라 시스템 전체에 걸쳐 분산되어 있는 개념이다.

4. 결론

정보통신 응용소프트웨어의 한 분야인 화상회의 시스템을 개발하는데 있어서 대두된 다양한 구현기법상 문제들에 대해, 본 논문에서는 기존의 클라이언트-서버 환경을 객체구현간의 직접 데이타 전송으로 해결하고자 하였다. 분산 시스템에서 가장 핵심적인 문제로 나타나는 원격지 시스템간의 효율적인 작업분담과 전송을 위한 방법론으로 제시된 객체구현 기반 개발방법은, 상기한바대로 각 객체구현들이 어플리케이션 도메인에 대한 정의를 전제로 개발되어야 한다. 이러한 어플리케이션의 범주를 점차적으로 넓혀 감으로써 범용의 기능을 갖는 객체 구현에 대한 연구가 차후에도 지속되어야 할 과제이다.

참 고 문 헌

[1] Adle Goldbeg and David Robson,"SMALLTALK - 80 - THE LANGUAGE and ITS IMPLEMENTATION", Addison Wesley Publishng Compnay.

[2] Bjarne Stroutstrup, "THE C++ PROGRAMMING LANGUAGE",Addison-Wesley Publishing Company.

[3] Brad J.Cox,"Object-Oriented Programming - An Evolution Approach",Addison-Wesley Publishing Company.

[4] Andrzej Goscinski " Distributed Operating Systems The Logical Design " Addison-Wesley Publishing Company , 1992 Reprinted

[5] Object Management Group(OMG),"Orbeline",OMG