분산 트레이더를 지원하는 경량(lightweight) 객체 모델 설계 및 구현 방안 연구

진 명 숙† · 송 병 권††

요 약

본 논문은 이동망을 포함하는 이질의 분산 컴퓨팅 환경에서 분산 트레이더의 개발을 위한 새로운 객체 모델인 LOM (Lightweight Object Model)과 이를 이용한 분산 트레이더의 구현 방안을 제시한다. 트레이더는 서버와 서비스가 동적으로 변하는 분산 환경에서, 서버에 관한 정보를 클라이언트에 제공함으로써, 클라이언트와 서버를 중개해 주는 역할을 한다. 트레이딩 서비스의 경우, 응용 시스템의 다양한 요구를 수용하는 복잡한 타입과 구조의 객체 모델의 도입보다는 트레이딩 서비스의 특성 및 분산 환경, 성능을 고려한 특성화된 객체 모델의 도입이 필요하다.

LOM은 새로운 참조 속성을 활용하여 객체간의 관계와 상속, 복합 속성 등 다양한 객체 지향 모델의 특성들을 수용하도록 하였다. 이를 통해 LOM은 주소 등 단순 데이터의 빠른 처리를 요구하는 트레이딩 서비스의 특성에 맞는 단순(간결)한 구조를 가진다. 또한 객체에 대한 권한 정보, 접근 제어 정보 등 보안에 관한 사항을 모델링 단계에서 포함하도록 하여 이동망에서의 신뢰성 보장과 트레이딩 정보의 분산 수용에 대처하도록 하였다. 본 논문에서는 LOM 모델과 함께 계산(시스템) 뷰점의 OMG (Object Management Group) 객체 모델과 정보 뷰점의 LOM을 연계시켜 분산 트레이더의 구현 방안을 제시한다.

A Study on the Design and Implementation of the Lightweight Object Model Supporting Distributed Trader

Myung-Sook Jin† · Byung-Kwen Song††

ABSTRACT

This paper presents a new object model, LOM (Lightweight Object Model) and an implementation method for the distributed trader in heterogeneous distributed computing environment including mobile network. Trader is a third party object that enables clients to find suitable servers, which provide the most appropriate services to clients in distributed environment including dynamic reconfiguration of services and servers. Trading service requires simpler and more specific object model than generic object models which provide richer multimedia data types and semantic characteristics with complex data structures.

LOM supports a new reference attribute type instead of the relationship, inheritance and composite attribute types of the general object oriented models and so LOM has simple data structures. Also in LOM, the modelling step includes specifying of the information about users and the access rights to objects for security in the mobile environment and development of the distributed storage for trading service. Also, we propose an implementation method of the distributed trader, which integrates the LOM-information object model and the OMG (object Management Group) computational object model.

† 신 회 원 : 경인여자대학 멀티미디어정보전산학부 교수
†† 종신회원 : 시경대학교 정보통신학과 교수
논문접수 : 1999년 9월 3일, 심사완료 : 2000년 3월 6일

1. 서 론

유·무선 통신망의 보급과 컴퓨터와 네트워크 장비 등 하드웨어의 발전에 따라 이동망을 포함한 분산 환경에서의 다양한 응용 시스템의 개발과 사용이 가속화되고 있으며 이와 병행하여 분산 환경에서의 개방(open) 시스템의 개발과 사용을 위한 표준화 활동이 활발히 이루어지고 있다[1, 19].

분산 컴퓨팅과 관련한 표준화 활동으로는 RM-ODP (the Reference Model of Open Distributed Processing), OSF/DCE(Open Software Foundation/Distributed Computing Environment), OMG/OMA(Object Management Group/Object Management Architecture), TINA-C/TINA(Telecommunications Information Networking Architecture-Consortium/TINA) 등이 있다. 이들은 분산 컴퓨팅을 위한 하부 구조의 제시와 하부 구조상에서 동작하는 다양한 분산 시스템들의 표준화된 모델 및 명세의 제시에 목적을 두고 있다[1, 12-15].

이들 중 RM-ODP는 개방 분산 처리(ODP) 시스템이 되기 위해 갖추어야 하는 개략적인 특징, 정의 및 (공통의) 제약에 대한 기반을 제공한다 RM-ODP는 분산 시스템에 대해 기업(enterprise), 정보(information), 계산(computing), 기술(technology), 공학(engineering)의 5가지로 분리된 추상화된 관점(Viewpoints)을 제시하고 있으며 OMG, TINA-C 등 타 표준화 활동에서도 이들 관점에 따라 분산 시스템을 기술하는 추세이다[1, 11, 14, 15]. 다섯 관점 중 응용 시스템의 개발과 사용에 있어서의 주요 관점은 정보, 계산 관점이다. 시스템에서 다루어지는 정보에 중점을 둔 정보 관점은 실세계의 대상을 모델링하는 다양한 데이터 타입과 데이터 간의 관계 등 세만틱(semantic)을 통한 데이터의 표현과 데이터의 저장, 관리 등에 중점을 둔다. 계산 관점은 분산 시스템의 동작, 즉 기능에 중점을 둔 관점으로, 분산 시스템을 특정 기능을 제공하기 위해 상호 작용하는 객체, 즉 계산 객체들의 집합으로 간주한다. 계산 관점에서는 계산 객체를 통해 분산 시스템의 구조, 인터페이스, 상호작용 등을 논리적으로 명세한다. 따라서 이들 두 관점에서의 기술(description)을 통해 분산 응용 시스템의 개발이 이루어질 수 있다

대표적인 계산 객체의 명세를 위한 언어로 분산 시스템 인터페이스의 기술에 널리 활용되고 있는 OMG IDL(Interface Definition Language)을 들 수 있다[2-4, 12] OMG IDL은 응용 시스템의 외부 인터페이스의 명세를 통해 시스템을 기술하는 언어로 분산 시스템의 내부 운영이나 관리에 대한 정의나 명세를 따로 두지 않고 있어 시스템마다 독자적인 개발이 가능하다[12]. 하지만 이러한 독자적인 시스템의 기술은 체계적이고 일관된 시스템 개발을 어렵게 하여 시스템의 확장 및 분산화에 걸림돌이 될 수 있다 또한 내부 데이터에 대한 정보 모델의 독자적인 구현이 시스템 외부 인터페이스의 명세(계산 명세)와 체계적으로 연계되지 못할 수 있다.

분산 객체 모델 등 분산 시스템 기술의 발전과 함께 최근 분산 환경에서의 응용 서비스로 관심을 모아지고 있는 분야 중의 하나가 트레이딩(trading) 서비스이다. 트레이딩 서비스는 다양한 특성을 갖는 서버가 존재하는 분산 환경에서, 특정 서비스를 이용하고자 하는 클라이언트가 해당 서버의 주소, 전달 파라메타, 서비스 품질(QOS : Quality Of Service) 등 서버에 관한 사전 지식이 없더라도 원하는 서비스를 제공하는 가장 적절한 서버를 찾아주는 것이다. 트레이딩 서비스를 제공하는 서버를 트레이더(trader)라고 한다. 트레이딩 서비스는 분산 환경이 점점 복잡해지고 확장될수록 중요성이 부각되는 서비스로 OMG, ODP, INA 등 표준화 단체와 각 연구기관에서의 관심이 커지고 있으나 분산 시스템의 전반적인 기술이 아직 확립되지 않은 단계로 트레이딩 서비스 또한 기반 기능의 제공에 초점을 두고 연구가 진행되고 있다[2-4].

기반 기능에 대한 연구와 함께 복잡해지는 분산 환경에서의 트레이딩 서비스의 제공을 위한 방법으로 분산 트레이더에 대한 연구들이 제안되고 있지만 분산 트레이더의 개발을 위해 기존의 일반화된 객체 모델을 도입하는 것은 다소 오버헤드가 있다고 판단된다 기존의 객체 모델들은 복잡한 실세계 대상을 반영하는 모델링에 중점을 두어 성능면에서의 고려가 이루어지지 못하고 있으며 객체간의 다양한 관계의 표현이나 상속성의 지원은 객체들의 분산 관리에 어려움을 준다. 이외에도 분산 환경에서 고려해야 할 보안이나 접근 제어 등에 대한 고려가 일반 객체 모델에서는 미흡하며 이러한 고려는 전송 서비스의 신뢰가 떨어지며, 사용자의 이동이 빈번한 이동망 환경에서는 더욱 절실한 문제이다[2, 3]. 따라서 기존의 일반화된 모델의 직접적인 도입보다는 응용의 특성이나 보안을 포함한 환경 및 성능에 대한 고려가 필요하다.

본 논문에서는 분산 응용 시스템의 예로 트레이딩 서비스를 위한 경량(lightweight) 객체 모델인 LOM (Lightweight Object Model)을 제시하고 이를 이용한 분산 트레이더의 구현 방안을 제시한다.

제안 모델은 ODP 객체 모델을 기반으로 하며, 트레이딩 서비스의 특성에 맞는 단순한 구조를 가지면서 동시에 객체간의 관계와 상속, 복합관계 등의 다양한 객체 지향 데이터 모델링의 특성들을 수용하도록 한다. 또한 분산 및 이동망 환경에서의 데이터의 보안과 서비스에 대한 권한 부여를 위해 객체에 대한 인한 정보, 접근 제어 정보 등 보안과 관련한 정보를 객체 명세에 포함하도록 한다. LOM의 제시와 함께 본 논문에서는 OMG IDL을 사용한 트레이더의 인터페이스의 기술과 이를 LOM과 연계시켜 정보 모델과 계산 모델을 일관되게 통합한 트레이더의 개발 방안을 제시한다. 즉 정보 객체의 데이터 지향 관점과 계산 객체의 처리 지향 관점을 연계시킨 시스템 지향의 통합 시스템 개발 모델을 제시한다. 이를 위해 정보 객체를 계산 객체의 일부분으로 흡수하며 계산 객체의 내부 구조 및 관리 방법의 동질화(homogeneous)를 통해 분산 시스템 개발 및 관리, 확장의 복잡성을 줄이도록 하였다.

본 논문의 구성은 2 절에서 관련 연구로 트레이더에 대한 연구와 기존의 정보, 계산 객체 모델을 살펴보고 3 절에서 LOM 모델을 기술하며 이를 계산 관점에서 확장한 분산 시스템 개발 모델을 제안한다. 4 절에서는 이의 구현 방안을 제시하며 5 절에서 결론을 맺는다.

2. 관련 연구

2.1 객체 모델

2.1.1 정보 모델

분산 시스템에서의 정보란 시스템의 각 요소들 간, 또는 시스템과 사용자가 간에 주고받는 데이터 및 시스템 자체가 가지는 지식(knowledge)을 일컫는 것으로, 정보는 시스템의 구현 방법이나 기술(technology)과는 독립되며, 정보 명세(information specification)를 통해 기술(description)된다. 정보 객체는 정보 명세에서 사용되는 단위 객체로, 각 객체들을 구별하게 하는 고유의 식별자(identity)를 가지며 객체를 포함하는 환경이나 객체들간의 영향 또는 자체적인 내부 활동에 의해 변화하는 상태(state)를 가진다. 정보 객체의 경우도 일반 객체 지향 패러다임에서 적용되는 타입, 클래스, 상속, 수퍼타입(supertype)과 서브타입(subtype), 객체 템플릿(template)의 개념이 적용된다[1, 20].

① ODP 정보 모델

ODP 정보 모델은 기본적인 객체 모델의 개념 외에 추가된 개념을 제안하며 대표적인 예가 불변 스키마(invariant schema), 동적 스키마(dynamic schema), 정적 스키마(static schema)이다. 불변 스키마는 객체를 클래스로 그룹짓는 방법이나 객체 타입에 대한 일반적인 기술 등 객체의 일관되게 변하지 않는 사항들을 정의하며, 동적 스키마는 객체 생성, 소멸, 그룹핑, 제그룹핑 등 불변 스키마를 따르는 객체의 행동에 대한 규칙을 기술하는 스키마이다. 정적 스키마는 특정 시점의 객체의 정적인 상태를 기술하는 스키마이다. ODP 정보 모델은 정보 객체에 대한 기본 개념을 제공하여 타 모델의 참고 모델로 도입되고 있다[1].

② OSI MIM(Management Information Model)

OSI MIM은 기본적인 객체 모델의 개념 외에 추가된 개념으로 ODP에서 다루지 못했던 관리/피관리 객체 관계를 포함한다. 피관리 객체는 관리 객체에 의해 제어되는 객체이다. 또한 복합(composition)관계의 특수한 형태인 포함(containment) 관계를 포함한다. 복합관계는 서브타입의 객체가 한 개 이상의 수퍼타입 객체의 부분 컴포넌트로 포함되는 데 반해 포함관계는 수퍼타입의 정확히 한 객체(인스턴스)의 컴포넌트가 되는 관계이다[15].

③ OSI GRM(General Relationship Model)

OSI GRM은 기본적인 객체 모델의 개념 외에 클래스간이나 객체간의 바인딩을 위한 관계(relationship)에 대한 상세한 정의를 포함한다. 관계의 정의를 위해 관계에 참여하는 객체의 수(cardinality)와 참여/탈퇴 규칙 등 특성을 포함하는 역할(role)과 관계에 참여하는 객체들의 추가, 제거, 참여 객체의 변경 등을 포함하는 행동(behaviour)으로 정의되는 관계 클래스 템플릿을 사용한다. 객체간의 관계에 대한 모델링은 TINA(Telecommunications Information Networking Architecture)에서도 고려하고 있는 사항으로 TINA에서는 한 역할에 대해 한 개 객체가 대응되어야 하나 GRM은 일대다 매핑이 가능하도록 하고 있어 좀 더 유동적인 모델로 볼 수 있다[15].

④ ODMG 객체 모델

ODMG(Object Database Management Group) 모델은 데이터베이스 지향의 객체 모델로 분산 환경에 대한 고려는 없으나 모델링 대상에 대한 다양한 시맨틱의 제공이나 키(key), 객체의 검색을 위한 질의의 제공 등은 향후 분산 시스템의 정보 모델에서도 지원되어야 할 부분이다 ODMG 객체 모델은 특성 타입의 모든 인스탄스 객체의 집합을 영역(extent)으로 정의하며 타입 선언 시 영역절을 포함하면 자동으로 영역 내의 각 객체에 대한 색인을 유지시킨다 또한 하나의 특성이나 다수 특성의 집합을 키(key)로 정의할 수 있다 그룹(collections) 형태로 집합(set), 백(bag), 리스트(list), 배열(array)을 지원한다 ODMG 객체는 명세 언어인 ODL(Object Definition Language)을 사용하여 명세된다[11].

2.2.2 계산 모델

계산 모델은 RM-ODP의 계산 모델이 주요 기반 모델이 되고 있으며, 분산 응용 시스템의 역할 수행에 중점을 둔 모델로 분산 응용 시스템은 인터페이스를 통해 상호 작용하는 계산 객체의 집합으로 간주된다 객체는 클라이언트에 제공할 하나 이상의 서비스를 캡슐화한 것으로 각 객체는 한 개이상의 인터페이스를 가지며, 이는 오퍼레이션 인터페이스와 연속미디어의 스트림 인터페이스로 나누어진다. 이러한 기능적인 인터페이스 대신 QOS와 같은 비기능적인 특성은 인터페이스의 서비스 속성(attribute)으로 취급되며 기능적인 측면과 속성은 인터페이스 템플릿에 의해 명세된다. 공통의 특성을 갖는 객체들의 집합을 빌딩 블록(building block)으로 정의한 그룹 개념을 도입하여 보안, 관리, 배치의 단위로 사용하기도 한다[1, 12]

계산 객체 모델 중 현재 널리 활용되는 모델로 OMG 객체 모델이 있다. OMG 객체 모델은 RM-ODP 모델을 수용하고 있는 ODP 모델의 부분으로 간주된다. 서비스 및 인터페이스를 제공하기 위한 계산 관점의 모델링을 위한 계산 언어인 OMG IDL(Interface Definition Language)을 제공함으로써 응용 시스템 개발의 기반이 되고 있다[12]

2.2 트레이딩 서비스

2.2.1 기본 개념

서버와 클라이언트가 분산되어 존재하는 분산 환경에서 클라이언트와 서버간의 바인딩은 해결해야할 중요 문제로 이러한 문제의 해결을 위해 트레이더를 둔다. 트레이더는 클라이언트와 서버를 중재해주는 객체로서 서버 객체가 트레이더에 등록한 서버에 대한 정보를 클라이언트에 제공한다. 서버는 자신이 제공하는 서비스의 종류 및 비 기능적인 특성 그리고 클라이언트가 바인드할 수 있는 위치에 대한 정보를 트레이더에 등록한다. 클라이언트는 트레이더에 서비스를 요청할 때 원하는 서비스의 종류 및 부가적인 요구 조건을 알려주게 된다. 트레이더는 클라이언트가 요구하는 조건에 맞는 가장 적절한 서버를 찾아 그 위치를 포함한 서버 정보를 클라이언트에 제공한다. 따라서 트레이더는 분산 환경에서 클라이어트와 서버가 그 위치에 관계없이 바인드될 수 있도록 함으로써 위치 문제를 해결해 주며, 많은 서버 객체 중 클라이언트가 적절한 서버에 바인드될 수 있도록 서버의 선택을 도와준다[2-4]

트레이더에 서버가 정보를 등록하는 과정을 익스포트(export), 클라이언트가 서버에 대해 정보를 요청하는 과정을 임포트(import)라고 하며 익스포트되는 서버의 정보를 서비스 오퍼(service offer)라고 한다. 한 개 트레이더는 자신이 관리하는 서버와 클라이언트의 집합을 가지며 이는 트레이딩 영역(trading domain)이 된다 분산 환경에서는 많은 트레이딩 영역이 있으며 트레이더는 서로 다른 트레이딩 영역간의 상호작용을 통해 서비스를 수행할 수 있다. 이러한 트레이더들간의 상호작용을 트레이더 연합(federation)이라 한다[2, 3].

2.2.2 트레이더의 정보 관점

정보 관점의 트레이더는 서비스 및 서비스를 제공하는 서버에 관한 정보를 저장, 관리하며 이 때 트레이더가 다루는 정보의 단위가 서비스 오퍼이다. 하나의 서비스 오퍼에는 특정 서비스에 대한 정보(서버 주소, 서비스 품질 등)가 들어있다[18]. 트레이더에서의 서비스, 서비스 오퍼, 컨텍스트 개념은 다음과 같다

① 서비스

서비스는 인터페이스의 서비스 속성으로 이루어지며, 인터페이스 시그너쳐(signature) 타입과 속성 타입으로 각각 기술된다. 같은 인터페이스 시그너쳐 타입을 가지면 같은 서비스로 볼 수 있으나 서비스 속성인 응답 지연 시간, 트랜스포트 프로토콜, 사용료, QOS

등 비 기능적인 측면에서 다를 수 있다. 서비스 속성은 시간에 따라 변하지 않는 정적 속성과 큐의 길이와 같이 시스템 운용 중에 변화하는 동적 속성으로 구성된다[18].

② 서비스 오퍼

서비스 오퍼는 트레이드 되는 서비스 정보로서 서비스 오퍼 식별자, 서비스 식별자, 서비스 인터페이스, 서비스 속성 값, 오퍼 만료일 등의 속성을 갖는다[18, 19].

③ 컨텍스트

컨텍스트는 서비스 오퍼 공간을 그룹으로 분류 구조화한 것이다. 그룹 속성으로 그룹 자체 속성과 서비스 오퍼의 공통 속성, 서비스의 공통 속성 등을 가진다[18, 19].

2.2.3 트레이더의 계산 관점

트레이더의 계산 관점은 트레이더의 인터페이스에 대한 기술로 볼 수 있다. ODP 트레이더 표준안에서는 OMG IDL로 트레이더를 기술하고 있으며, 서비스 인터페이스로 임포트, 엑스포트, 관리 인터페이스로 연합 간 링크 관리, 속성 관리 등의 인터페이스를 가진다[17, 18, 22].

2.2.4 트레이더 관련 연구 현황

분산 환경에서 제공되는 서비스에 대한 정보를 클라이언트에게 알려주는 트레이딩 서비스는 분산 환경에서 차지하는 트레이딩 서비스의 비중으로 인해, 유선망 환경에서의 트레이더에 대한 연구는 비교적 활발히 추진되고 있다. OMG 등 표준화 활동에서 기본 트레이딩 서비스의 모델링과 시스템 구조를 제시하고 있으며 이를 바탕으로 여러 연구기관에서의 CORBA, X.500 디렉토리 시스템을 활용한 구현 방법들을 제시하고 있다. 표준화 단체들의 연구에서는 서비스 오퍼의 증가에 대한 대책으로 일정 규칙으로 서비스 오퍼를 분류하여 연결 트리(tree)로 표시한 논리적인 구조인 컨텍스트 구조를 제시하고 있으며 학교 및 연구소를 중심으로 서비스 영역 분할 방안과 분산 트레이더의 개발에 대한 연구가 추진되고 있다[]. 이처럼 대부분의 트레이더 관련 연구가 기능의 제공에 중점을 두고 있으며 구체적인 성능 향상 방안이나 보안, 데이터 모델과 같은 세부적인 연구가 제대로 이루어지지 못하고 있다[2-4, 16].

3. 제안 객체 모델

본 절에서는 분산 환경에서 기반 분산 응용 시스템인 트레이더를 위한 객체 모델인 LOM을 제시한다.

제안 객체 모델은 기본적으로 RM-ODP의 정보 계산 모델을 수용하며, 분산 트레이딩 서비스에 적합한 객체 모델의 제시를 목적으로 한다. 제안 객체 모델은 기존 객체 모델에 비해 경량(lightweight) 객체 모델로 다음과 같은 점들을 주요 고려 사항으로 하여 설계하였다.

① 제안 모델은 ODP 객체 모델을 기반으로 하며, 트레이딩 서비스의 특성에 맞는 단순한 구조를 가지면서 동시에 객체간의 관계와 상속, 복합관계 등의 다양한 객체 지향 데이터 모델링의 특성들을 수용하도록 한다

② 분산 및 이동망 환경에서의 데이터의 보안과 서비스에 대한 권한 부여를 위해 객체에 대한 권한 정보, 접근 제어 정보 등 보안과 관련한 정보를 객체 명세에 포함하도록 한다.

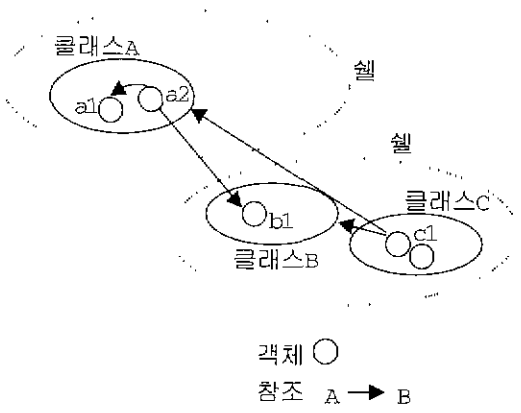
③ 분산 응용 시스템의 개발을 위해 OMG IDL을 사용한 계산 인터페이스 명세가 일반화되는 추세이다. 따라서 IDL을 사용한 계산 객체의 기술과 연계시켜 LOM 정보 모델과 OMG 계산 모델을 일관되게 통합한 응용 시스템 개발을 위한 시스템 모델을 제시한다. 즉 정보 객체의 데이터 지향 관점과 계산 객체의 처리 지향 관점을 연계시킨 시스템 지향의 객체 모델을 제시한다.

3.1 LOM 정보 모델

트레이딩 서비스의 서비스 오퍼의 모델링을 위한 LOM 정보 모델은 ODP 정보 모델을 수용하면서 다음의 주요 특성들을 갖는다.

3.1.1 정보 객체와 클래스, 쉘

제안 모델의 정보 객체는 트레이딩 서비스의 서비스 오퍼에 적용할 수 있는 개념으로 스키마에 해당되는 다수 속성(attribute)들과 해당 속성 값들로 구성된다. 정보 관점에서 제어(control)의 기본 단위는 독립된 정보 객체이며 정보 객체 클래스는 같은 타입을 갖는 정보 객체들의 집합으로 정의되며, 서비스 오퍼 컨텍스트에 적용될 수 있다.



(그림 1) 객체, 클래스, 쉘

효율적인 분산 환경의 지원을 위해, LOM에서는 객체들의 클래스를 물리적 위치에 따라 그룹으로 나누어 각 그룹을 쉘(shell)로 정한다. 외부 쉘로부터 접근될 수 있는 클래스를 외부(external) 클래스, 쉘 내부의 클래스에 의해서만 접근될 수 있는 클래스를 내부(internal) 클래스, 소유자에 의해서만 접근되는 클래스를 독립(independent) 클래스로 구분한다. (그림 1)에서 클래스A는 외부 클래스이며, 클래스B는 내부 클래스, 클래스C은 독립 클래스이다. 객체에 대해서도 자신이 속한 클래스 내·외에서의 접근 여부에 따라 내부 객체와 외부 객체로 구분한다. 쉘과 클래스, 정보 객체뿐만 아니라 객체의 각 속성 및 속성 값에 대해서도 독립된 제어가 가능하다.

(그림 2)는 부록의 LOM 정보 객체 명세 사양의 일부를 나타낸 것으로 객체 및 클래스를 정의하는 방법을 보여준다. 객체 외에 클래스도 고유의 속성을 가지며, 클래스와 객체 모두 권한(3.1.3절에서 기술)에 대한 정보로 클래스 권한인 <class_right>, 객체 권한인 <information_object_right_type>을 갖는다

```
<information_class_template> ::= [PERSIST]CLASS
                        <class_name> <class_right>
                        (<class_property>) (<class_method>)
<information_object_template> = OBJECT<class_name>
                        <information_object_right_type>
                        (<information_object_property>)
                        (<information_object_method>)
```

(그림 2) 정보 객체 명세

3.1.2 속성 타입

정보 객체의 속성들이 가지는 타입(부록 참조)은 정수, 실수, 문자, 문자열 이외에도 더 정보 객체를 가리키는 객체 참조 타입, 클래스 전체를 가리키는 클래스 타입의 참조 속성이 있다. 또한 한가지 타입을 갖는 속성뿐만 아니라 재귀적으로 구성된 속성의 집합을 단일 속성으로 하는 복합 속성(composite attribute)을 가질 수 있다

객체 지향 모델의 기본적인 상속 개념은 재사용성 측면에서 유용한 개념이나 수퍼타입 객체에서 발생되는 문제들이 이를 상속하는 서브타입 객체로 파급되며, 객체가 물리적으로 분산 관리되는 경우 수퍼/서브 관계의 지속적인 유지에 많은 어려움이 있다. 따라서 LOM에서는 상속 대신 참조 속성을 이에 대신하는 개념으로 도입하였다.

LOM의 참조 속성은 모두 4가지로 객체의 참조의 경우 자신이 속한 클래스와 타 클래스의 객체를 참조하는 IOBJ(Internal OBJect), EOBJ(External OBJect) 타입이 있으며, 클래스 참조의 경우 자신이 속한 쉘, 내부의 클래스와 외부 클래스를 참조하는 ICLASS(Internal CLASS), ECLASS(External CLASS)를 가진다 (<표 1>)

<표 1> 참조 속성

타 입	설 명
IOBJ(Internal OBJect pointer)	같은 클래스 내의 객체 참조
EOBJ(External OBJect pointer)	타 클래스의 객체 참조
ICLASS(Internal CLASS pointer)	같은 쉘 내의 클래스 참조
ECLASS(External CLASS pointer)	쉘 외부의 클래스 참조

(그림 3)은 LOM의 복합 속성인 COMP, 객체 참조 속성인 EOBJ 속성을 사용한 객체 및 클래스 명세의 예를 보여준다. (그림 3)의 SERVIEC_OFFER 객체의 명세에서 서비스 오퍼 식별자(svcOfferID)와 서비스 오퍼 속성들(svcOfferProp1, ... svcOfferPropN)과 서비스에 대한 속성인 svcDesc를 포함한다. 서비스 오퍼에 포함된 서비스와 관련한 속성은 복합 속성인 COMP로 표현함으로써 서비스 속성이 타 일반 오퍼 속성들과 구별되는 내포 관계를 명시할 수 있다. 이는 (그림 3)의 (b)와 같이 SERVICE 클래스가 따로 존재할 경우, SERVICE

클래스의 한 객체를 참조하도록 객체 참조 속성인 EOBJ 타입을 사용하여 대체할 수 있어 분산 환경의 상황에 따라 독립적인 시스템의 구성이 가능하다.

```
PERSIST CLASS SERVICE_OFFER 666 (numOfOffer, INT)
OBJECT SERVICE_OFFER RIGHT
      ((svcOfferID INT, PRIMARY_KEY),
      (svcDesc, COMP((svcDescID, INT)
      (interfaceID, INT), (prop1, LSTR),
      (prop2, LSTR)  , (propN, SSTR),
      (compatibleSvc, EOBJ SERVICE)),
      (svcOfferProp1, INT), ... (svcOfferPropN, SSTR))

CLASS SERVICE 644 (numOfService, INT)
OBJECT SERVICE RIGHT ((svcID, INT),
      (svcProp1, INT), (svcProp2, SSTR),  ,
      (svcPropN, SSTR), (compatibleSvc, EOBJ SERVICE))
```

(a)

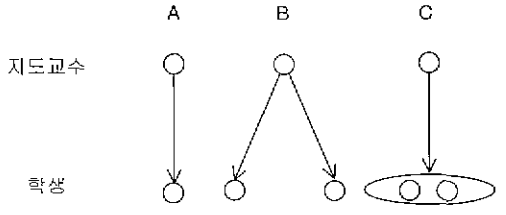
```
PERSIST CLASS SERVICE_OFFER 666
    (numOfOffer, INT)
OBJECT SERVICE_OFFER RIGHT
    ((svcOfferID, INT, PRIMARY_KEY),
    (svcDesc EOBJ SERVICE),
    (svcOfferProp1, INT) ... (svcOfferPropN, SSTR))
```

(b)

(그림 3) 정보 객체 명세 예

제안 모델에서는 일반 객체 모델에서와 같은 객체간의 관계에 대한 명세를 따로 정의하지 않고 참조 속성을 사용하여 객체의 명세와 동일한 방법으로 명세하며 일반 객체와 같은 방식으로 관리된다

(그림 4)는 일반적인 객체간의 관계를 나타낸다. (그림 4)의 (A)는 일대일 관계, (B)는 일대일 관계를 변형한 일대다 관계, (C)는 OSI-GRM에서 지원되는 일대다 관계를 보여준다.



(그림 4) 객체간의 관계

(그림 5)는 LOM 에서의 관계에 대한 명세의 예로 CLASS1 객체와 CLASS2 객체와의 관계인 REL_C1_C2를 명세한 것으로 EOBJ 타입을 사용하여 CLASS1과 CLASS2를 연계하고 있으며 관계에 관한 속성 rProp1, rProp2를 포함한다

(그림 5)의 (a) 명세는 (그림 4)의 (A)에 해당되는 일대일 관계이며 (그림 5)의 (b)는 (그림 4)의 (C)와 같은 관계를 나타낸다. (그림 5)의 (c)는 LOM이 관계의 대상에 대해 OR연산이 가능함을 보여준다.

```
PERSIST CLASS REL_C1_C2 666
OBJECT REL_C1_C2 RIGHT ((role1 EOBJ CLASS1_OBJ),
    (role2, EOBJ CLASS2_OBJ), (rProp1, INT), (rProp2, SSTR))
```

(a)

```
PERSIST CLASS REL_C1_C2 666
OBJECT REL_C1_C2 RIGHT ((role1, EOBJ CLASS1_OBJ),
    (role2, SET, EOBJ CLASS2_OBJ), (rProp1, INT), (rProp2, SSTR))
```

(b)

```
PERSIST CLASS REL_C1_C2 666
OBJECT REL_C1_C2C3 RIGHT ((role1, EOBJ CLASS1_OBJ)
    (role2, OR, EOBJ CLASS2_OBJ, EOBJ CLASS3_OBJ),
    (rProp1 INT) (rProp2, SSTR))
```

(c)

(그림 5) 객체간의 관계에 대한 명세

3.1.3 권한(클래스, 객체)

LOM에서는 정보 객체 또는 클래스 단위의 접근 제어를 수행한다. UNIX 시스템의 파일에 대한 사용자 접근 제어와 같이 객체의 생성자를 소유자로 하여 소유자, 소유자와 같은 그룹, 타 그룹 사용자 등으로 구분하여 읽기, 쓰기, 실행의 분리된 권리를 가진다. (그림 3)의 클래스 SERVICE_OFFER 명세에서 666은 소유자, 그룹, 타 사용자가 모두 SERVICE_OFFER 클래스에 대해서는 읽기, 쓰기의 권한을 가지는 것을 의미한다. SERVICE 클래스의 644는 소유자는 읽기, 쓰기, 실행을 그룹 및 타 사용자는 읽기, 쓰기 권한을 가지는 것을 의미한다. 객체의 경우 객체 인스탄스의 생성시 클래스의 경우와 같이 객체별로 권한을 정의한다. 이러한 객체와 클래스에 대한 사용자의 권한의 명세는 응용에서의 이를 활용한 사용자 접근 제어를 수행할 수 있도록 한다. 예를 들어 클래스 명세의 변경은 쓰기 권한이, 값의 참조는 읽기 권한이 요구되도록 할 수 있다(<표 2>)

〈표 2〉 서비스에 따른 사용자의 권리 분류

구분	서비스의 종류	사용자의 권리
속성	속성 추가, 제거, 이름 변경	클래스에 대해 'write' 권리
정보 객체	정보 객체 값의 추가, 제거, 변경 정보 객체 값의 검색	정보 객체에 대해 'write' 권리 정보 객체에 대해 'read' 권리
사용자	사용자 이름의 등록, 삭제	클래스 소유자와 같은 그룹

3.2 계산 모델

3.2.1 계산 객체 구조

정보 관점의 LOM을 바탕으로 제안한 계산 관점의 객체 모델에서의 객체의 구조는 <표 3>과 같다. 계산 객체는 지속 특성(persistent property), 일시 특성(transient property), 외부 메쏘드(external methods), 내부 메쏘드(internal Methods)의 네 부분으로 구성되며 모든 계산 객체는 이러한 등질의 정립된 구조를 가진다

〈표 3〉 계산 객체 구조

구 분		설 명	구 현
특성	지속 특성 (persistent property)	계산 객체의 비 기능적인 저장 속성	정보객체(클래스)
	일시 특성 (transient property)	계산 객체의 비 기능적인 일시 속성	OMG IDL을 통한 기술
메쏘드	외부 메쏘드 (external methods)	계산 객체 인터페이스	OMG IDL을 통한 기술
	내부 메쏘드 (internal Methods)	계산 객체 관리 모듈	사용자 정의

특성(property)은 계산 객체에서 사용되는 정적인 데이터를 기술한다. 이는 계산 객체의 라이프 사이클과 무관하게 지속적으로 저장, 관리되는 1) 지속 특성(persistent property)과 계산 객체와 수명을 같이하여 계산 객체의 라이프 사이클 동안에 생성되고 소멸되는 2) 일시 특성(transient property)으로 나누어진다 LOM 정보 객체 모델이 계산 객체의 지속 특성의 모델로 도입된다.

메쏘드는 계산 객체의 동적인 행위를 기술한 것으로 내·외부 메쏘드로 나누어진다 본 논문에서는 연속 미디어 스트림을 고려하지 않으니 기존의 오퍼레이션이나 스트림 인터페이스와 구분된 통합 개념으로 메쏘드를 도입하였다 내부 메쏘드는 외부 계산 객체에 보여지지 않는 계산 객체 내부에서의 행위를 기술하며 외부 메쏘드는 타 계산 객체에 보여지는 인터페이스를 구성하는 함수이다. 내부 메쏘드의 예로는 정보 객체의 타임 관리, 저장 관리 등을 들 수 있으며 외부 메쏘드 지원 기능을 수행하는 함수들로 구성된다. 외부 메쏘드는 OMG IDL로 기술되며 외부 계산 객체에 보여지는 인터페이스로 구성된다.

3.2.2 외부 메쏘드 : 트레이더 외부 인터페이스

트레이더 외부 인터페이스는 임포트와 익스포트를 포함한 서비스 인터페이스와 트레이더 연합간의 링크 관리를 포함한 관리 인터페이스로 크게 나누어진다. ODP 트레이더 표준안에서는 이를 OMG IDL로 기술하고 있다. 제안 모델에서 트레이더 외부 인터페이스는 계산 객체의 외부 메쏘드로 구분된다. 본 논문에서는 표준안에서 다루지 않은 컨텍스트에 관한 인터페이스를 (그림 6)과 같이 추가로 명세한다

```
Module CTX{
    //type definition
        ..... .....
    //interface
    interface ctxUserIf{    //context user interface

    void addSvcOffer(           //add service offer
        in IdentifierType       clntId,
        in ContextNameType      ctxName,
        in ServiceOfferType     svcOffer
        out IdentifierType      svcOfferId
    ) raise(OP_FAILURE),

    void deleteSvcOffer(        //delete service offer
        in IdentifierType       clntId,
        in IdentifierType       svcOfferId
    ) raise(OP_FAILURE),

    void modifySvcOffer(        //modify service offer
        in IdentifierType       clntId,
        in ServiceOfferType     svcOffer,
        in IdentifierType       svcOfferId
    ) raise(OP_FAILURE),

};
```

(그림 6) OMG IDL을 사용한 트레이딩 서비스의 부분적 명세

3.2.3 내부 메쏘드 트레이더 내부 관리 함수

계산 객체는 타 계산 객체에 보여지는 인터페이스인 외부 메쏘드 외에 자체적인 내부 메쏘드를 가진다. 내부 메쏘드는 계산 객체 내부의 관리 기능과 하부 환경에 대한 제어 기능을 수행하고 이를 외부 메쏘드에 제공한다 내부 메쏘드의 예로는 정보 객체들의 타임 관리, 저장 관리 및 검색 관리, 제어 기능 등을 들 수 있다

다음은 정보 객체의 타입 관리를 위한 내부 메쏘드로 속성 추가와 제거 함수와 정보객체 관리 함수인 정보객체 생성 및 제거 함수이다.

① 속성 추가(addAttribute)

사용자와 그룹 암호 입력 후 권한을 확인한다. 제거될 속성 일뭉을 입력받아 속성이 이미 존재하는지 여부를 확인한다. 새로운 속성 테이블을 할당받아 기존의 테이블 리스트에 연결한다.

② 속성 제거(delAttribute)

사용자와 그룹암호 입력 후 확인한다. 제거될 속성 이름을 입력받아 속성이 존재하는지 여부를 확인한다. 해당 속성 테이블을 가리키는 연결 리스트 포인터를, 해당 속성의 다음 속성 테이블을 가리키도록 변경함으로써 해당 속성 테이블을 참조할 수 없도록 한다

③ 객체 생성(createObject)

사용자와 그룹 암호를 입력받아 사용자가 객체 생성 권한이 있는지 확인한다 속성 테이블을 참조하여 객체 테이블과 객체 저장 영역을 할당받고 이에 속성 테이블의 타입에 따른 값을 입력받아 저장한다.

④ 객체 제거(deleteObject)

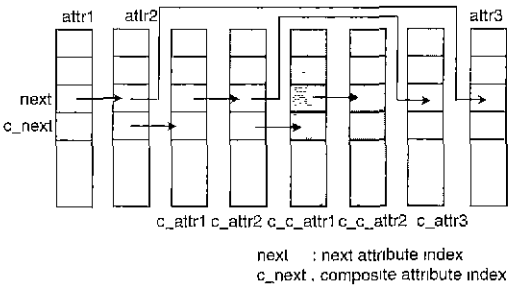
사용자와 그룹 암호를 입력받아 사용자가 객체 제거 권한이 있는지 확인한다 할당된 객체 테이블과 저장 영역을 반환한다.

4. 분산 트레이더 구현 방안

본 절에서는 제안 계산 모델의 내부 메쏘드로 정의된 정보객체의 관리 기능의 구현 방안을 객체 타입관리, 사용자 등록을 포함한 접근 제어, 객체 저장관리의 세부 항목으로 나누어 기술한다.

4.1 타입 관리

객체의 타입 관리를 위해 객체 속성의 제거와 추가, 속성의 이름 변경이 가능해야 한다. 트레이딩 정보의 변경에 따라 계산 객체가 활성화 된 후에도 동적인 속성 제어가 가능해야 하며, 객체 속성뿐만 아니라 클래스 속성도 객체 속성과 같이 유지되고 관리되어야 한다. 이를 위해 각 속성 당 하나씩 속성 테이블을 할당할 수 있으며 속성 테이블은 각 속성의 이름과 타입, 속성값의 위치(offset), 길이, 다음 속성테이블의 주소를 유지하여야 한다 LOM의 복합 속성 타입(COMP)을 지원하기 위해서는 (그림 7)과 같은 연결 리스트 구조를 갖는 테이블에 복합 속성을 위한 포인터를 두어 다중 내포 형태의 복합 속성을 관리할 수 있다.



(그림 7) 복합 속성 지원을 위한 속성 테이블 구조

4.2 정보 객체 관리

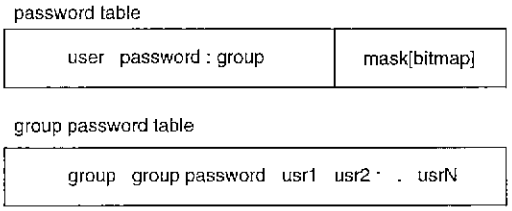
LOM 정보 객체는 객체의 각 속성 타입에 맞는 값을 입력하여 객체 저장소에 저장함으로써 생겨난다. 트레이더의 엑스포트와 임포트 인터페이스를 지원하는 내부 함수로서 정보객체의 생성과 내용 검색, 제거, 값의 변경을 수행하는 함수가 있어야 한다. 속성 데이블과 마찬가지로 객체 저장소와 함께 정보객체의 관리를 위한 정보, 즉 저장소 내의 객체의 위치, 객체의 크기, 객체에 대한 접근제어 정보 등을 유지할 테이블(정보객체 테이블)이 필요하다.

4.3 접근 제어(access control)

클래스와 정보 객체 단위의 접근 제어를 위해, 정보객체에 대한 사용자의 권리를 속성과 정보 객체 값의 변경 및 검색, 사용자 암호의 등록과 삭제 등 서비스에 따라 차별화 한다(<표 2>) 이를 위해 정보 객체 클래스 별로 사용자의 정보를 저장하는 사용자 암호(password) 테이블과 그룹 암호(group password) 테이블을 둔다. 사용자 암호 테이블에는 사용자 식별자와 비밀번호, 속한 그룹 식별자 외에 속성에 대한 속성별 마스크(mask)를 두어 속성별 접근제어가 가능하도록 할 수 있다(그림 8).

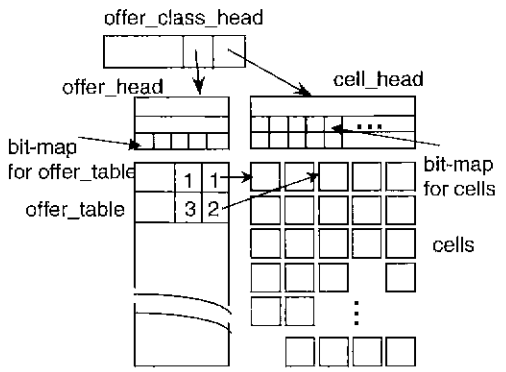
4.4 정보 객체 저장 관리

서비스 오퍼는 정형화된 데이터 타입을 가지며 데이



(그림 8) 접근 제어 정보 테이블

터의 크기도 비교적 적다 따라서 간단하고 확장과 분산이 용이한 저장 구조를 통해 성능을 향상시킬 수 있다. 본 논문에서는 정보 객체(서비스 오퍼)의 저장을 위해 정보 객체 길이에 해당되는 다수의 셀(cell)들을 할당받아 이에 정보 객체 값을 저장하는 방안을 제안한다. 이 때 셀은 정보 객체 저장을 위한 단위 크기(32바이트)의 저장 영역이며 효율적인 셀의 관리를 위해 셀 헤드 테이블(cell_head)을 둔다 셀 헤드 테이블 내에 할당된 셀과 프리 상태의 셀의 구분을 위해 셀 단위의 비트맵 정보를 두어 손쉬운 정보 객체의 저장 및 참조, 제거가 가능하도록 한다. (그림 9)는 객체 테이블(object_table)과 이를 관리하는 객체 헤드 테이블(object_head_table), 셀 리스트와 이를 관리하는 셀 헤드 테이블(cell_head_table) 간의 관계를 보여준다.



(그림 9) 서비스 오퍼 저장 구조

5. 결 론

본 논문에서는 이동망을 포함하는 이질의 분산 컴퓨팅 환경에서 분산 트레이더의 개발을 위한 새로운 객체 모델인 LOM(Lightweight Object Model)을 제안하고 이를 이용한 분산 트레이더의 구현 방안을 제시하였다.

제안된 모델은 ODP 객체 모델을 기반으로 하며, 트레이딩 서비스의 특성에 맞는 단순한 구조를 가지면서 동시에 참조 속성을 통해 객체간의 관계와 상속, 복합 관계 등의 다양한 객체 지향 데이터 모델링의 특성들을 수용하도록 하였다. 또한 분산 및 이동망 환경에서의 데이터의 보안과 서비스에 대한 권한 부여를 위해 객체에 대한 권한 정보, 접근 제어 정보 등 보안과 관련한 정보를 객체 명세에 포함하도록 하였다.

최근 분산 응용 시스템의 개발을 위해 OMG IDL을 사용한 계산 인터페이스 명세가 일반화되는 추세이다. 따라서 정보관점 모델인 LOM과 OMG 계산 모델을 일관되게 통합한 응용 시스템 개발을 위한 시스템 모델을 제시하고 트레이더의 구현 방안을 제안하였다. 즉 정보 객체의 데이터 지향 관점과 처리 객체의 처리 지향 관점을 연계시킨 시스템 지향의 모델을 통해 계산 객체의 구조 및 관리 방법을 동질화(homogeneous)하여 분산 시스템 관리의 복잡성을 줄일 수 있도록 하였다.

부 록

정보 모델 명세(BNF)

```
<information_class_template>  = [PERSIST]CLASS <class_name> <class_right>
                              (<class_property>) (<class_method>)
<information_object_template> = OBJECT <class_name>
                               <information_object_right_type>
                              (<information_object_property>)
                              (<information_object_method>)
<class_property>              ::= <property_list>
<class_method>                ::= <method_signature_list>
<information_object_property> ::= <property_list>
<property_list>               ::= <property>
                              | <property_list> <property>
<property>                    ::= (NAME , <type>)
<type>                        ::= <simple_type_data>
                              | <complex_type_data>
                              | <information_object_pointer_type_data>
                              | <class_pointer_type_data>
                              | <OR_type_data>
                              | <SET_type_data>
<simple_type_data>            ::= <simple_type>
                              [PRIMARY KEY | SECONDARY KEY |
                               MANDATORY KEY]
<complex_type_data>           ::= <complex_type> ( <property_list> )
<information_object_pointer_type_data>  = <information_object_pointer_type>
                              [PRIMARY KEY | SECONDARY KEY |
                               MANDATORY KEY]
<class_pointer_type_data>     = <class_pointer_type> [MANDATORY KEY]
<simple_type>                 ::= INT | REAL | SSTR | LSTR | BIN | TEXT
<composite_type>              = COMP
<information_object_pointer_type>  = (IOB | EOB) <class_name>
```

```
<class_pointer_type>                  = ICLASS | ECLASS
<OR_type_data>                 ::= OR , <type_name> , <type_name>
<SET_type_data>                ::= SET , <type_name>
<type_name>                    ::= <parameter_type>
<class_name>                   ::= <identifier>
<information_object_method>    = <method_signature_list>
<method_signature_list>                 ::= <method_signature>
                               | <method_signature_list> <method_signature>
<method_signature>                      ::= (<method_type> <identifier>
                                            ( <parameter_list> ))
<parameter_list>               ::= <parameter>
                               | <parameter_list> , <parameter>
<parameter>                    ::= <parameter_type> <identifier>
<method_type>                  ::= <parameter_type>
                               | void
<parameter_type>               ::= <simple_type>
                               | <complex_type>
                               | <information_object_pointer_type>
                               | <class_pointer_type>
<class_right>                   = <owner_right><group_right><other_right>
<owner_right>                   = <right>
<group_right>                   = <right>
<other_right>                   = <right>
<right>                        ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
<information_object_right_type> ::= RIGHT
```

SSTR : short string
LSTR : long string
IOBJ : internal information object
EOBJ : external information object
ICLASS : internal class
ECLASS : external class
0 ~ 7 : right bitmap

참 고 문 헌

[1] ISO/IEC JTC1/SC21 N7065 : ISO Working Draft on Basic Reference Model of Open Distributed Processing-Part 3 : Prescriptive Model, 1995.

[2] Mirion Bearman, "Trading in Open Distributed Environments, Proc. of the International Conference on Open Distributed Processing." Brisbane, Australia February 1995.

[3] Mueller-Jones, M Merz and W Lamersdorf, "The TRADEr Integrating Trading into DCE, Proc. of the International Conference on Open Distributed Processing," Brisbane, Australia, February 1995.

[4] Wilson Cheng and Xiaohua Jia, "A Reliable Trading Service in Open Distributed Systems, Proceedings of the First International Workshop on Services in Distributed and Networked Environments (SDNE), June 1994

[5] Andreas Vogel, Brigitte Kerherve, Gregor Von Bochmann and Jan Gecsei, Distributed Multimedia and QOS : A Survey, IEEE Multimedia, Vol.2, No.2, Summer, 1995.

[6] The Common Object Request Broker : Architecture and Specification, OMG Document 93.XX.YY, Revision 1.2, December 1993.

[7] C. Aurrecoechea, A. Campbell and L Hauw, A Review of Quality of Service Architectures, ACM Multimedia Systems Journal, November, 1995

[8] M. S Jin, Y J. Kim, S. S An, An Information and Computational Model for Distributed Multimedia Computing, KISS Journal, November, 1997.

[10] G G Cattell, The Object Database Standard : ODMG-93 Release 1.1.

[11] The Common Object Request Broker : Architecture and Specification, OMG Document 93.XX.YY, Revision 1.2, December 1993

[12] Common Facilities Architecture, OMG Document 95_1_2, Revision 4.0, January, 1995

[13] Object Services Architecture, OMG Document, Revision 8.1, January, 1995

[14] TINA DPE Information Modelling Concepts, TB_EAC.001_1.1_93, November, 1994.

[15] TINA DPE Computational Modelling Concepts, TB_A2.HC.012_1.2_94, February, 1995

[16] Claudia Popien and Bernd Meyer, Federating ODP Traders : An X.500 Approach, Proceedings of the International Conference on Communications 1993 (ICC'93), Geneva, Switzerland, pp.313-317.

[17] Claudia Popien and Bernd Meyer, Object Configuration by ODP Traders, Proceedings of the IFIP TC6/WG6.1 International Conference on Open Distributed Processing, Berlin, Germany, 13-16 September, 1993, Publ. North-Holland, pp.406-408

[18] Claudia Popien and Miguel Heineken, Trading Enhancement by Service Combination in ODP, Proceedings of the IFIP TC6/WG6.1 International Conference on Open Distributed Processing, Berlin, Germany, 13-16 September, 1993, Publ North-Holland, pp.384-386.

[19] ISO/IEC 10746-3, "Open Distributed Processing-Reference Model-Part 3 Architecture," 1995.

[20] ISO/IEC JTC1/SC21, "Draft Rec. X.950-ISO/IEC 13235-1-ODP Trading Function : Specification, 2nd DIS, 23 January, 1997.

[21] ISO/IEC JTC1/SC21, "Rec.X Str/Draft ODP Trading Function ISO/IEC 13235 . 1994/Draft ODP Trading Function," 29 July, 1994.

진 명 숙

e-mail . jinms@dove.kyungin-c.ac.kr

1990년 고려대학교 전자공학과 졸업(공학사)

1992년 고려대학교 대학원 전자공학과(공학석사)

1997년 고려대학교 대학원 전자공학과(공학박사)

1996년~현재 경인여자대학 멀티미디어정보전산학부 조교수

관심분야 . 분산 시스템, 멀티미디어 컴퓨팅, 객체 지향 시스템

송 병 권

e-mail : bksong@bukak.sookyeong.ac.kr

1984년 고려대학교 전자공학과 졸업(공학사)

1986년 고려대학교 대학원 전자공학과(공학석사)

1995년 고려대학교 대학원 전자공학과(공학박사)

1984년~1997년 삼성종합기술원 선임연구원

1995년~현재 서경대학교 정보통신공학과 교수

관심분야 : High-Speed Network, 분산처리시스템, Mobile Computing