CORBA 환경에서 실시간 응용 지원을 위한 분산 객체그룹 플랫폼의 설계 및 구현

김 명 희[†] · 이 재 완[††] · 주 수 종[†††]

요 약

분산 객체 컴퓨팅 환경에서 실행되는 어플리케이션들은 서비스를 지원하기 위해 분산되어 있는 객체들을 관리해야 하는 어려움을 초래한다. 또한, 비디오나 오디오와 같은 멀티미디어 서비스를 위한 어플리케이션들은 실시간 제약 조건을 만족해야 하기 때문에, 사용자들은 분산 멀티미디어 서비스들에 실시간 메커니즘을 적용할 필요성을 느낀다. 본 논문의 목적은 분산된 객체들의 관리와 더불어 실시간 지원 어플리케이션의 개발을 용이하게 하는데 있다. 이를 위해, 어플리케이션과 CORBA 사이에 실시간 지원 객체그룹 플랫폼을 설계하고 구현하였다. 이 플랫폼은 기존의 객체그룹 모델[13, 14]에 실시간 지원에 관련된 스케줄러와 타이머 객체등을 추가하여 확장하였다. 이 플랫폼의 구성요소들을 설계하기 위해서 객체모델, 기능모델, 동적모델 등의 객체 모델링[6] 기술을 이용하여 요구사항들을 분석한 후, IDL을 이용하여 구성요소들의 세부 인터페이스를 기술하였으며, IONA의 OrbixMT 2.2를 사용하여 실시간 객체그룹의 플랫폼을 구현하였다. 마지막으로, 실시간 객체그룹 플랫폼 상에서 실시간 객체그룹 내 구성요소들중 스케줄러 객체의 실행과정을 보였다

A Design and Implementation of Distributed Object Group Platform for Supporting Real-Time Application in CORBA Environments

Myung-Hee Kim[†] · Jae-Wan Lee[††] · Su-Chong Joo[†††]

ABSTRACT

The applications developing in distributed object computing environments are faced with the difficulties for managing various lots of distributed objects. Also, because the most multimedia services, like video, audio, and so forth, must be satisfied itself with real-time constraints, the users also are feeling with necessary to apply real-time mechanisms to distributed multimedia services. The goal of this paper is to solve the problems for managing distributed objects, and to be easy to develop complex applications that can provide real-time services. To do this, we designed and implemented a real-time object group platform that can be placed between applications and CORBA. This platform is extended the *existing object group model[13, 14] added to the scheduler* and timer object components for supporting real-time concept. We designed the components for platform by using James Rumbaugh object modeling technology that consists of object, function and dynamic model. And then we described the detailed interfaces of the components by IDL and implemented our real-time object group's platform using OrbixMT 2.2 which is the IONA Technologies' ORB product. Finally, we showed the execution procedures of the scheduler object of each components in a real-time object group platform.

※ 본 논문은 1997년도 한국학술진흥 재단의 공모과제 연구비에 의하여 연구되었음

† 준 회 원 : 원광대학교 대학원 컴퓨터공학과
†† 정 회 원 : 군산대학교 전자정보공학부 교수
††† 정 회 원 : 원광대학교 컴퓨터공학과 교수

논문접수 : 1999년 9월 1일, 심사완료 : 2000년 2월 7일

1. 서 론

근래의 컴퓨팅 환경은 사용자들에게 물리적인 위치에 상관없이 다양하고 신속한 서비스를 제공할 수 있는 분산환경으로 발전하고 있으며, 이에 따른 절차지향적 서비스 방법이 점차 객체지향적 방법으로 변화되고 있다. 이러한 분산 컴퓨팅 환경에 객체지향 기술이 도입되면서 분산 객체 컴퓨팅 분야가 대두되고 있다. 그러나, 분산 환경에서 어플리케이션 개발자가 분산 어플리케이션을 설계할 때, 네트워크 전반에 걸친 데이터의 상호교환에 따른 프로토콜 정의 등의 상호작용의 설계에 많은 시간을 소비한다. 또한 각기 다른 프로토콜의 개념과 규약으로 인한 분산 어플리케이션의 호환이 불가능하다는 문제점이 발생한다 이러한 문제점으로 인해서, 분산 환경의 표준화 구조인 OMG의 CORBA[1]가 정의되었고, 네트워크 전역에 광범위의 서비스를 제공하기 위한 통신망에 기반을 둔 분산 객체지향 구조의 설계와 기술을 위한 TINA[2]가 TINA-C에서 제시되었다. CORBA와 TINA는 특정한 인터페이스의 다중 구현을 지원하고, 클라이언트에게 서버의 위치 투명성을 제공하며, 이종의 광역 네트워크 기반으로 동일 도메인의 분산 객체 환경을 정의하는 유사성을 가진다. 반면에, 객체의 개념과 분산 환경 그리고 서비스의 관점에서 차이점[2]을 가지고 있는 분산 객체 컴퓨팅 환경의 표준화 구조들이다.

이러한 분산 환경 하에서 복잡한 분산 소프트웨어의 개발 및 관리의 복잡성을 줄이고 분산된 객체들을 효율적으로 관리하기 위한 객체들의 모임인 객체그룹의 정의가 필요하다. TINA에서 이미 객체그룹에 대한 정의가 제안되었으나, 객체그룹에 대한 구조나 구성요소들의 기능정립이 아닌 개념적 정의의 기술이었다. 이에 대해, 본 연구의 선행 연구[13, 14]로서 객체그룹에 대한 모델링과 구조제시를 CORBA를 기반으로 수행했다.

최근까지 분산객체 기술의 표준안인 CORBA를 기반으로 한 연구들은 일반적인 데이터 전송의 정확성과 효율성에 초점을 맞추어 진행해왔다 그러나, 최근의 경향은 연속적인 미디어(Continuous media)가 컴퓨터 네트워크에 적용되기 시작하면서 미디어에 따른 시간적 특성의 보장을 요구하게 되었다 이로 인하여, OMG에서는 RT-SIG(Realtime-Special Interest Group)[3]를 발족하여, 기존의 표준 CORBA에 실시간 요구사항을 적용시킬수 있는 표준안 작성을 진행하고 있다 또한, 이와 관련된 개별적인 연구들[4, 5]은 CORBA의 핵심이 되는 ORB를 수정하거나 확장하여 실시간을 지원하도록 하는 방향으로 초점이 맞추어져 이루어지고있다

따라서, 본 논문에서는 객체들의 효율적인 관리와 분산 소프트웨어 개발을 용이하게 하는 객체그룹의 개념을 기본 구조로 하여, 기존의 표준 CORBA상에서 실시간을 지원하는, 실시간 분산 어플리케이션의 하부구조인 실시간 객체그룹 플랫폼을 제시하여 모델링하고 구현한다. 본 논문에서 제시된 플랫폼은 기존의 표준 CORBA에서도 특정한 실시간 운영체제에 의존하지 않으며, 객체들을 효율적으로 관리하고 분산 소프트웨어의 개발이 용이하며, 실시간 어플리케이션을 지원할 수 있는 장점을 제공한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구와 관련된 연구들과 실시간 지원을 위해 본 논문에서 사용하는 스케쥴링 기법 등에 대해 간략하게 설명하고, 3장은 실시간 객체그룹 플랫폼 구조를 객체, 동적, 기능 모델로 나누어지는 James Rambaugh[6]기법으로 모델링하고, 4장은 실시간 객체그룹 플랫폼 구조와 구현 환경, 그리고 구현과정과 구성요소들중 스케쥴러 객체의 실행과정을 보인다. 마지막으로 결론과 향후 연구과제를 제시한다.

2. 관련연구

본 절에서는 실시간 객체그룹 플랫폼과 연관된 객체지향 미들웨어인 CORBA와 실시간 CORBA에 대하여 기술하고, 실시간 지원을 위한 전역 시간 서비스와 서비스를 수행을 위한 실시간 스케쥴링 방법에 대해서 기술한다.

2.1 CORBA와 실시간 CORBA

분산 시스템 또는 분산 어플리케이션을 개발하기 위해서는 운영체제와 네트웍 환경 등 이기종(Heterogeneous)의 환경 하에서 작동 가능한 클라이언트/서버 소프트웨어를 개발해야 할뿐만 아니라 개발된 소프트웨어는 분산환경에서 이기종의 시스템과도 쉽게 통합할 수 있어야 한다. 이러한 문제들을 해결하며 분산환경에서 객체지향 응용 프로그램의 재사용성(Reusability), 상호운용성(Interoperability)과 이식성(Portability)을 위한 공통 프레임웍 환경을 제공하도록 표준안으로 제안된 것이 OMG에서 제정한 CORBA이다. CORBA는 OMG

에서 제안한 OMA(Object Management Architecture) 표준의 일부이다. OMA는 응용 프로그램간의 결합뿐만 아니라 객체의 생성, 소멸에서부터 저장, 트랜잭션 기능에 이르기까지 분산환경 하에서 필요로 하는 전반적인 객체 솔루션을 제공한다. 이에 반해 CORBA는 OMA의 일부분으로써 주로 분산 환경 하에서 객체간의 통신기능과 이를 조작하는데 필요한 기능들을 주로 언급하고 있다. 마찬가지로 ORB(Object Request Broker)는 객체의 위치나 활동 그리고 서비스 요구 및 수신간 연결의 투명성을 제공함으로써 CORBA 표준안에서 주로 통신 부분만을 담당하고 있다.

실시간 CORBA 시스템이란, CORBA 시스템에서 종단간 실행하는데 실시간 조건을 만족하는 시스템을 말한다. 즉, 표준 CORBA에 실시간 개념을 포함한 것을 의미한다. CORBA는 시간 제약을 가지는 실시간 응용 프로그램을 지원하기에는 제한 사항들이 많기 때문에 OMG의 RT-SIG에서는 표준 CORBA에 실시간 지원을 위한 기능을 추가하거나 확장하는 작업을 진행[4, 5]하고 있다. 여기에서는 실시간 CORBA를 위해서 CORBA 시스템에서 종단간 실행에서 실시간 규약의 표현과 시행을 하고 있으며, 결과적으로 1996년 표준 CORBA를 실시간 CORBA로 확장할 경우에 필요한 기술과 개념을 정의하기 위하여 실시간 CORBA 백서(White Paper)를 발표[3]하였는데, 여기에서는 기본적인 실시간 개념인 시간 제약, 스케쥴링, QoS, 성능, 동기/비동기적 분산 시스템 모델, 결함허용 등을 정의하고 있다. 또한, 실시간 CORBA에 관련된 여러 RFP가 발표되었는데, 실시간 ORB를 위한 기술로 고정 우선순위 스케쥴링과 종단간 예측성을 위한 ORB 자원관리, 융통성을 가진 통신 등을 권고하고 있다. 세계적으로 실시간 CORBA를 연구하여 개발중인 제품으로는 워싱턴 대학에서 개발한 고성능의 표준 CORBA와 호환되는 실시간 ORB인 TAO와 Rhode Island 대학과 미 해군 NRaD(US Navy Research and Development Laboratories), 그리고 MITRE사가 개발한 NRaD는 다중스레드를 지원하며, CORBA 시스템 내의 동적 종단간 시간제약을 표현하며 클라이언트나 서버가 추가 또는 삭제되고, 시간 제약이 변경될 수 있는 환경을 제공하며 이외에 Chorus 사의 CHORUS/COOL 등 여러 제품들이 있다.

그러나, 위의 연구들은 ORB를 수정하거나 확장하여 실시간을 지원하는 CORBA들이므로, 본 연구는 ORB를 확장하지 않고 CORBA를 기반으로 하여 또 다른 응용 플랫폼으로써 객체그룹을 위치시킴으로써 실시간 어플리케이션을 지원할 수 있도록 한다. 이를 위해, 전역 시간 서비스, 실시간 스케쥴링 서비스를 수행할 수 있는 새로운 객체들을 추가하여 기존의 객체그룹[13, 14]을 확장시켰다.

2.2 전역 시간 서비스

분산 컴퓨팅 환경에서 클라이언트와 서브는 현재 전역 시간을 얻기 위해 타임 서비의 오피레이션을 호출하여, 각 시스템을 시간편차(skew)내에서 동기화시켜야 한다. CORBA는 ORB를 기반으로 한 분산환경 하에서 특정 시간 순서에 따른 이벤트 처리를 위해 이들 시간을 동기화 시키는 서비스가 필요함에 따라, OMG에서 COSS 3(Common Object Service Specification 3)에 객체 시간 서비스의 명세를 규정[9]하고 있다. 객체 시간 서비스는 DCE(Distributed Computing Environment) 타임 서비스에서 정의하고 있는 UTC(Universal Time Coordinated)를 기본적인 시간 표현방법으로 사용한다. 여기에서 규정한 객체 시간 서비스는 크게 기본 시간 서비스와 타이머 이벤트 서비스로 나누어진다.

기본 시간 서비스는 어플리케이션 등의 클라이언트에 대해서, 시간의 취득, 시간의 에러폭의 취득, 두 가지 시간의 비교, 두 가지 시간의 시간 간격 계산, 두 가지 시간간격의 오버랩 시간 계산등의 기능을 제공한다.

타이머 이벤트 서비스는 시간을 설정 혹은 해제하기 위한 인터페이스와 이벤트를 등록 혹은 등록을 해제하기 위한 인터페이스로 구성되어 있다. 시간에는 절대시간, 상대시간, 혹은 주기적인 시간의 어느 쪽을 설정할 수 있다. 여기서 설정된 시각이 되면 등록된 이벤트가 이벤트 채널을 경유하여 어플리케이션으로 송신된다. 또한, 이벤트의 송신에는 이벤트 서비스의 푸시 모델을 사용하고 있다.

실시간 시스템에서는 시간 조건을 표현하고, 시행하며 시간 조건의 위반을 조절하는 기능이 필요하다. 실시간 CORBA 시스템에서 서버에서 클라이언트의 메소드(method) 호출은 시간 조건 동작의 한 예이다. 대부분 시간 조건 표현은 시작 시간, 마감시간, 동작 주기로 표현된다. 시작 시간 조건은 초기 시작 시간(Earliest start time) 조건과 마지막 시작 시간(Latest start time) 조건으로 나뉜다. 즉, 만약 동작이 지정된 시간에 시작되지 않았다면, 에러가 발생한다. 마지막 시작 시간은 실제적으로 발생되기 전 일어날 수 있는 마감시간 위반

이나 계획된 스케줄의 가능한 위반을 검출하는데 유용하다. 마감시간은 동작이 완료되어야 할 때까지의 시간을 나타낸다. 주기 조건은 초기 시작 시간과 동작의 반복된 경우에 대한 규칙적인 시간 간격으로 시작 시간과 마감시간을 표현한다[3]. 시간 조건은 정적으로 설계 시나, 동적으로 실행 시에 지정될 수 있으며, 절대 시간이나 상대시간으로 표현된다. 절대 시간은 시스템의 전역 시간을 기반으로 한다. 또한 절대 시간은 절대 시간을 유지하는 서버로부터 얻기나, 아니면 분산적인 방법으로 얻을 수 있다. 상대 시간은 요구 시점에서부터 측정한 상대적인 시간을 말한다

2.3 서비스 수행을 위한 실시간 스케쥴링

실시간 시스템에서 스케쥴링[12]은 메소드 호출과 연관된 작업의 상대적인 실행 위치를 기술하는 우선 순위(priority)를 나타내므로 중요하다. 대부분의 실시간 스케쥴링 알고리즘은 우선 순위를 기본으로 하고 그 시스템이나 환경 또는 어플리케이션에 맞는 우선 순위 할당방법을 사용한다. 스케쥴링은 예측성을 포함하는 성능 측정 방법을 갖는다 스케쥴링 알고리즘의 성능은 기대하는 스케쥴링 최적화 제약을 얼마나 잘 충족되었는지에 의해 결정된다. 따라서, 실시간 CORBA 시스템은 집합적인 완료 시간의 예측이 필요하다 이 예측성은 어떤 특별한 어플리케이션에 대한 자원의 할당과 재할당을 관리하게 된다

실시간 시스템에서 사용하는 스케쥴링 기법[12]에는 크게, hard real-time과 soft real-time 스케쥴링, 우선 순위에 의한 분류로 선점형(Preemptive)과 비선점형(Non-preemptive) 스케쥴링, 중앙 집중식과 비중앙 집중식 스케쥴링, 동적 스케쥴링과 정적 스케쥴링 기법이 있다. 이와에도 주기 및 비주기 태스크에 대한 마감시간 만족을 위한 스케쥴링 기법[16] 등이 연구되어 왔다.

이러한 기법들 중에서, 동적 스케쥴링의 EDF(Earliest Deadline First) 스케쥴링 기법[10]은 새로운 작업이 도착할 때마다 현재 실행 중인 작업들과 새로운 작업의 마감시간을 서로 비교하여 새로운 작업의 우선 순위를 결정한다. 이 때 마감시간이 빠를수록 더 높은 우선 순위를 부여받게 된다. EDF 기법에서 스케줄 가능성 분석을 위한 조건을 제시하였는데 전체 프로세서 이용률 U를 구하는 식은 식 (1)과 같다.

$$U = \sum_{i=1}^{N} U_i = \sum_{i=1}^{N} \frac{C_i}{T_i} \leq 1 \qquad (1)$$

식 (1)에서 U_i는 프로세서의 이용률, T_i는 작업의 주기 그리고, C_i는 실행시간을 나타낸다 이 조건식은 필요충분조건으로서 정확한 스케줄 가능성 분석 방법을 제공하며, 이는 또한 EDF 기법이 우선 순위 기반 선점형 스케쥴링 방식 중에서 최적의 스케쥴링 방법임을 의미한다. 즉, EDF 기법에 의해서 스케쥴될 수 없는 태스크 집합은 다른 어떤 우선 순위 기반 스케쥴링 방법을 사용하더라도 스케쥴 될 수 없다.

따라서, 본 연구에서는 실시간 객체그룹을 지원하기 위한 시간 지원 서비스들로서, 전역 시간 서비스와 Soft Real-Time, 그리고 EDF에 의한 스케쥴링 기법을 사용하고자 한다.

본 연구에서 사용하는 타임 서비스는 다음과 같은 과정을 통하여 수행되고 스케쥴링된다 먼저 객체들의 동기화를 맞추기 위해 타이머 객체를 사용하며, 클라이언트가 서버에게 메시지 전송 전에 타이머로부터 전역 시간을 요청하여 얻어오는 Cristian의 알고리즘[11]을 기본으로 한다. 그러나, 이 알고리즘은 클라이언트와 타이머간, 그리고 클라이언트와 그룹관리자간의 전송 시간 동안의 시간이 존재한다. 따라서 본 연구에서는 이 알고리즘을 다음과 같이 수정하여 사용한다

클라이언트의 초기에 얻은 전역 시간이 T_0이고, 다음으로 타이머에게 요청하여 얻은 전역 시간이 T_1이라면 단 방향 메시지 전송 시간은 식 (2)과 같이 나타낸다

$$(T_1 - T_0)/2 \qquad (2)$$

또한 여기에 더 정밀하게 메시지 전송 시간을 얻기를 위해 타이머에서의 인터럽트 조작 시간 I를 알고 있다면 단방향의 전송 시간은 식 (3)과 같이 나타낸다.

$$(T_1 - T_0 - I)/2 \qquad (3)$$

식 (3)과 같이 얻어진 단방향 메시지 전송 시간은 타이머에게 요청한 클라이언트의 현재 시간에 추가하여 서비스를 요청하기 전의 전역 시간을 얻는다 또한, 이렇게 동기화 된 현재 시간과 마감시간을 서버에게 알려주면 서버는 클라이언트의 시간 T_c과 현재 서버의 시간 T_s을 비교하여 두 시스템간의 시간편차(skew)을 포함한 클라이언트에서 서버까지 메시지 전송 시간 T_a을 얻는 수식은 식 (4)와 같다.

$$T_a = T_s - T_c \qquad (4)$$

또한, 클라이언트의 메시지 전송 시간 T_g을 받은 서버에서 실제 사용되는 마감시간 T_e은 다음 식 (5)와 같다. 여기에서 Td는 클라이언트가 요청한 마감시간이다.

$$T_e = T_d - T_g \quad (5)$$

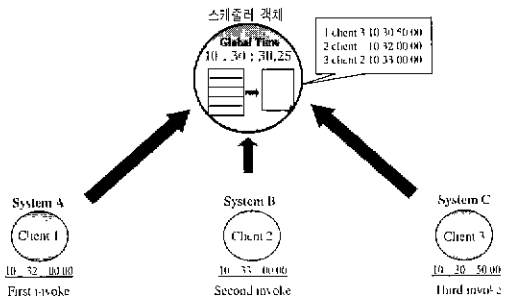
본 연구의 스케줄링 시에 우선순위를 식 (5)에서 구해진 서버에서 실제 사용되는 마감시간 T_e 값과 전역 시간을 비교함으로써 EDF 기법에 의해 높은 우선순위를 부여한다.

다음은 위의 식을 예를 들어 우선순위를 결정하는 과정을 설명한다. 현재 클라이언트의 시간 T_c가 10:30:25.25이고, 서버의 시간 T_s가 10:30:30.25이며, 서비스 실행시간이 30초인 객체에게 마감시간 T_d가 10:32:00:00으로 시스템 A가 서비스를 요청하였다고 가정하면, 서버에서 실제적 사용되는 마감시간 T_e은 다음과 같이 구해진다.

T_g = 10:30:30.25 - 10:30:25.25 = 5.00초,

T_e = 10:32:00.00 - 5.00 = 10:31:55:00

따라서, 서비스 객체의 실제적인 마감시간이 10:31:55.00이므로 이 시간까지 서버가 수행을 완료하여 결과를 클라이언트에게 반환하면 마감시간을 만족할 수 있다. (그림 1)은 위의 과정을 통한 스케줄 과정을 나타낸다.



(그림 1) 스케줄링 과정

위의 과정에서 각각 시스템 내에서 요청한 클라이언트들이 갖고 있는 시간은 서버 측의 스케줄러에게 식 (5)에 의해서 구해진 마감시간을 말한다. 이 마감시간에는 메시지 전송 시간을 포함하여 계산한 값들이다. 여기에서 시스템 C의 클라이언트 3이 마지막으로 요청하였더라도 마감시간이 가장 짧음으로 EDF 기법에 의해서 우선순위가 가장 높게 부여된다.

3. 실시간 객체그룹 플랫폼

본 장에서는 실시간을 지원하는 객체그룹의 플랫폼을 모델링하기 위해 필요한 요구사항들을 기술하고, 이를 토대로 실시간 지원 객체그룹의 구조와 구성요소들 그리고 그들의 기능에 관하여 기술한다.

3.1 요구사항

기존의 TINA에서 제시한 객체그룹은 관리와 서비스의 효율성을 위해 객체들의 집합체적인 단위로 정의되어있다. 그러나 앞에서 언급한 것처럼 TINA의 객체그룹은 단순히 정의에 대한 기술뿐이며, 완전한 모델이나 구조가 제시되어있지 않다. 따라서, 선행연구로서 TINA에서 제안한 객체그룹의 정의를 CORBA환경에 적용시켜 CORBA기반의 객체그룹 모델을 정의[13, 14]하였다. 따라서, 본 논문에서는 이를 확장하여 실시간 지원 객체그룹을 정의하였으며, 실시간 지원 객체그룹은 단순한 객체들의 모임이 아닌 특성을 가진 실시간 객체들의 집합으로 정의한다. 이때 객체의 특성이란 하나의 서비스를 수행하기 위한 마감시간을 갖는 특정 객체들의 집합으로 정의하고 있다. 즉, 하나의 서비스를 수행하는 일련의 객체들을 모아 그룹화시킨 것이다. 이러한 정의를 토대로 실시간 지원 객체그룹을 모델링하기 위해서는, 객체그룹의 분산성, 보안성, 실시간적 특성 등의 분산 시스템과 실시간 시스템의 특성을 고려하여야 하며, 고려에 따른 실시간 지원 객체그룹의 요구사항들은 다음과 같다.

- 객체그룹은 그룹관리자(GroupManager), 객체팩토리(ObjectFactory), 스케줄러 객체(Scheduler Object), 보안 객체(Security Object), 객체정보 레포지토리(ObjectInformation Repository), 객체(Object)등의 요소가 필요하다.
- 객체그룹내의 객체나 서브객체그룹들은 동일 도메인 내의 다른 시스템으로 분산 가능하다.
- 객체그룹은 그룹관리자를 두어 객체그룹내의 객체와 서브객체그룹들을 관리하는 기능을 가진다.
- 클라이언트는 객체그룹의 보안 관리를 위해서 그룹관리자를 통하여 객체그룹에 접속하여야 한다.
- 실시간 지원에 따른 서비스 수행의 예견성(Predic-

tability)을 위하여 그룹관리자가 객체의 생성 시에 서비스 객체들의 수행으로 서비스 실행시간을 얻는 과정이 필요하다

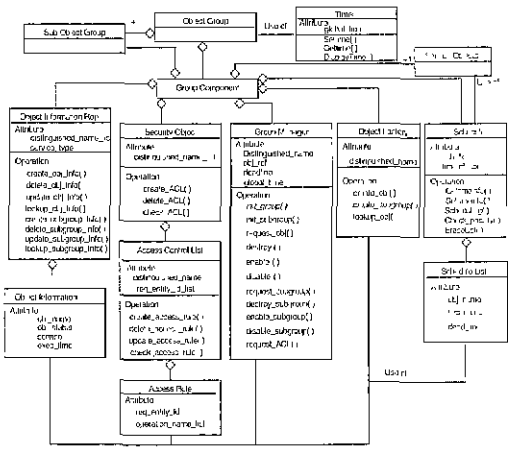
- 서비스 수행을 요구하는 클라이언트 객체들에 대한 우선순위 등의 실시간 서비스 지원에 따른 관리를 하는 스케쥴러 객체가 필요하다.
- 객체그룹과 객체들의 정보(실시간 지원 정보도 포함)를 저장하기 위해 객체정보 레포지토리가 필요하다.
- 객체팩토리는 그룹관리자의 객체생성 요구를 수행하기 위해 객체정보 레포지토리와 상호 작용하여 객체를 생성하여야 한다.
- 보안 객체를 두어 분산 객체 시스템이 필요로 하는 보안 기능 중 인증(Authentication)과 인가(Authorization)기능을 수행하도록 한다
- 객체그룹내부에 서브객체그룹이 존재하며, 객체그룹의 구조와 기능이 동일하다.
- 객체그룹은 여러 개의 서브객체그룹을 가지며, 깊이(Depth)가 1인 서브객체그룹을 가질 수 있다. 즉, 서브객체그룹은 하위 레벨이나 서브객체그룹을 내포할 수 없다.
- 절대적 시간과 상대적 시간의 표준 타입 명시가 필요하다.
- 전역 우선 순위의 지원을 하여야 한다
- 모든 CORBA 개체 서비스에서의 우선 순위 큐잉 지원이 있어야 한다.
- 고정 우선 순위로 스케줄 가능한 개체가 정의되어야 한다.
- 클라이언트가 요청에 관련된 시간 제한 정보에 대한 통보할 수 있는 기능과 인터페이스가 명시되어야 한다

3.2 실시간 객체그룹 모델링

본 절에서는 실시간 객체그룹 플랫폼을 위한 실시간 객체그룹 모델링 과정[6]으로, 시스템을 객체 혹은 서로 연관된 객체들로 파악하여 표현하는 객체모델, 객체의 호출 변환 관점의 기능 모델, 이벤트라는 개념에 의존하여 변화를 표현하는 동적 모델을 통하여 설계한다

3.2.1 객체 모델

그룹관리자(Group Manger)는 외부의 서비스 요청을 처리하고, 객체그룹과 연관된 내외의 모든 요구사항을 관리한다. 또한, 객체정보 레포지토리, 보안 객체, 스케줄러, 서비스 객체들과 상호작용을 하며, 객체그룹내의 모든 서비스 객체와 구성요소(객체정보 레포지토리, 스케줄러 등)들의 관리를 위한 기능 즉, 객체의 생성, 삭제, 활성화, 비활성화와 서브객체그룹의 그룹관리자와 구성요소들의 생성과 삭제, 서브객체그룹의 활성화와 비활성화 등을 수행한다. 객체정보 레포지토리(Object Information Repository)는 객체그룹 내의 객체들과 서브객체그룹들에 대한 정보(객체그룹내의 서비스 객체 생성을 위한 템플릿과 서비스 객체의 상태 등)를 관리한다. 객체팩토리(Object Factory)는 그룹관리자의 서비스 객체 생성 요구에 따라 객체정보 레포지토리에 있는 속성객체들의 정보를 이용하여 객체그룹의 모든 객체(서비스 객체 및 서브그룹관리자)를 생성한다. 보안(Security) 객체는 객체그룹 외부에서 객체그룹 내의 객체나 서브객체그룹의 객체에 대한 서비스 요청 시 보안 검사를 하고 그룹 내에 접근할 수 있도록, 이와 관련된 작업을 수행한다. 실시간 서비스 수행을 지원하기 위해, 스케줄러(Scheduler)는 서비스 요청된 객체그룹에 대한 실행시간 등의 정보를 이용하여 서비스 수행을 EDF (Earliest Deadline First)로 스케줄링 한다 여기에서 사용되는 시간은 구현 시스템마다 일치하지 않을 수 있으므로 서비스의 일관성을 위하여 타이머에서 시간 값을 얻어 사용하게 된다. 서비스 객체는 그룹관리자에 의해 관리되는 실질적인 서비스를 제공하는 객체이다. 서브객체그룹은 앞에서 언급한 객체그룹의 특성을 그대로 상속받은 객체그룹이다. 타이머(Timer)는 서로 다른 객체그룹간의 동기화를 위하여 대표적인



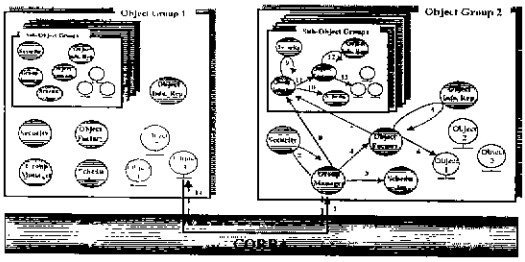
(그림 2) 실시간 객체그룹 클래스 다이아그램

시간 동기화 알고리즘인 Cristian 방법을 적용하였으며, 객체그룹의 외부에 위치한 클라이언트의 요구 시에 클라이언트 측과 서버 측에게 요구 시마다 수동적(passive)으로 전역시간을 알려주는 역할을 수행한다. 위에서 정의한 실시간 객체그룹 구성요소들의 클래스 다이아그램은 (그림 2)와 같다.

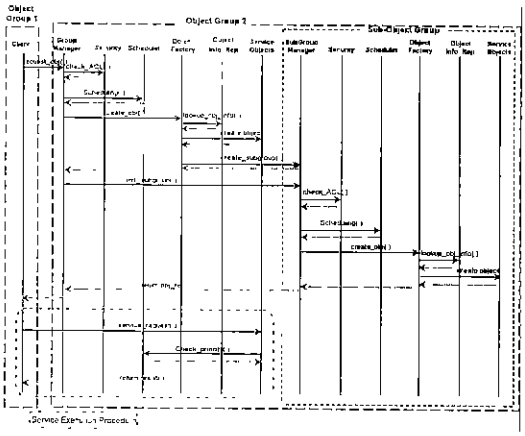
3.2.2 기능 및 동적 모델

분산되어 있는 객체들을 관리하기 위한 방안으로는 그룹관리자를 통하여 동일 객체그룹내와 타 객체그룹내의 서비스 객체나 서브객체그룹의 생성, 삭제, 활성화, 비활성화등의 기능을 수행한다. 즉, 동일 객체그룹의 서브객체그룹의 생성, 삭제, 활성화, 비활성화, 동일 객체그룹의 서브객체그룹내 서비스 객체의 생성, 삭제, 활성화 및 비활성화, 타 객체그룹의 서비스 객체의 생성, 삭제, 활성화 및 비활성화등의 관리기능을 수행한다. 이러한 관리방안중에, 다른 객체그룹 내의 클라이언트가 서비스 요청으로 서비스 수행을 하는 객체들의 생성과정을 예로 들어 설명하며, 이들간의 호출과정은 (그림 3)과 같다. 먼저 객체그룹1(OG1)의 객체가 특정 서비스 수행의 요구(객체와 서브객체그룹의 생성)를 객체그룹2(OG2)의 그룹관리자에게 요청하며, 그룹관리자는 보안객체에게 보안 검사를 의뢰하고, 인증을 받은 객체라면, 스케줄러에게 요청한 클라이언트의 레퍼런스와 마감시간을 넘겨준다. 스케쥴러는 그룹관리자로부터 전달받은 정보를 이용하여 클라이언트의 우선 순위를 결정하고, 해당 객체그룹이 초기화될 때 객체들의 수행 예측 시간을 유지한다. 그룹관리자는 객체팩토리에게 객체그룹의 구성요소들인 객체와 서브객체그룹의 그룹관리자 생성을 요청한다. 객체팩토리는 객체정보 레포지토리의 템플릿을 참조하여 객체와 서브그룹관리자를 생성하고, 서브객체그룹 내에 객체의 생성은 상위 그룹관리자가 서브그룹관리자에게 요청하여 이루어지며 생성과정은 상위 그룹과 같다. 객체팩토리 객체는 생성한 객체들의 정보를 객체정보 레포지토리에 저장하고, 생성된 객체들의 레퍼런스 중 서비스를 수행하기 위한 첫 번째 객체의 레퍼런스를 그룹관리자에게 반환하며, 그룹관리자는 다시 서비스를 요청한 객체에게 이 레퍼런스를 반환한다.

분산된 OG2 객체그룹과 서브 객체그룹 내에 서비스 객체의 생성 및 생성 후 서비스 요청 시(그림 3 참조), 하나의 클라이언트가 객체그룹에 접속 시 이벤트들의 변화를 ETD(Event Trace Diagram)으로 (그림 4)에 나타내었다.



(그림 3) 객체그룹 및 서브 객체그룹내의 서비스 객체 생성 및 서비스 접속 과정



(그림 4) 객체그룹 및 서브 객체그룹내의 서비스 객체 생성 및 서비스 요청에 대한 ETD

3.3 실시간 객체그룹 IDL 설계

본 절에서는 실시간 객체그룹의 구성 요소들 중에서 이전의 객체그룹 모델에서 기능의 차이를 보이는 그룹관리자 객체와 실시간 지원과 가장 밀접한 타이머와 스케줄러 객체의 IDL을 나타낸다. 그룹 관리자 객체는 객체그룹 내의 객체에 서비스를 요구하도록 객체그룹을 초기화하는 init_group()과 서비스 요구를 위한 request_obj(), 객체를 삭제, 활성화, 비활성화 시키기 위해 각각 destroy(), enable(), disable() 오퍼레이션 등을 제공한다. 또한 서브객체그룹을 위해서 init_subgroup(), request_subgroup(), destroy_subgroup(), enable_subgroup(), disable_subgroup() 오퍼레이션을 제공하며, 보안 검사를 위한 request_ACL()오퍼레이션을 제공한다. 여기에

서 init_group()오퍼레이션이 객체그룹을 미리 수행시켜 실시간의 예견성에 따른 객체그룹의 서비스 실행시간을 얻어낸다. 타이머 객체는 각 클라이언트 측과 서버 측의 상호 작용 전에 동기화를 위해서 전역시간 서비스를 제공하며, 속성으로써 전역시간을 가지며, 시간을 얻어오기 위한 Gettime()과 시간을 통보하는 Settime(), 시간을 표시하는 DisplayTime()오퍼레이션을 제공한다. 스케줄러 객체는 서비스 객체에 대한 스케줄 정보를 얻기 위한 GettimeInfo(), 시간 정보를 통보해주는 SettimeInfo(), 스케줄링을 위한 Scheduling(), 우선 순위를 통지해주기 위한 Check_priority(), 스케줄링 후에 리스트를 삭제하기 위한 EraseList() 오퍼레이션을 제공한다.

(그림 5)와 (그림 6) 그리고 (그림 7)은 각각 그룹관리자 객체와 타이머 객체 그리고 스케줄러 객체의 IDL이다.

```
interface Group_Manager {
  typedef Object              Object_Reference;

  // 그룹관리자 객체의 속성 //
  readonly attribute Object_Reference  obj_ref;
  readonly attribute DN                Distinguished_name;
  readonly attribute Deadline_Time     deadline;
  // 그룹의 초기화 서비스 //
  void init_group(in Template_Name group_template_name);
  // 객체 관리 서비스 //
  boolean request_obj(in Template_Name obj_template_name,
                      in Object_information obj_info, in
                      Deadline_Time deadline);
  void destroy(in DN distinguished_name, in
  Request_Entity_Id entity_id);
  void enable(in DN distinguished_name);
  void disable(in DN distinguished_name);
  // 서브객체그룹 내의 초기화 서비스 //
  void init_subgroup(in Template_Name
  subgroup_template_name);
  // 서브객체그룹 관리 서비스 //
  boolean request_subgroup(in Template_Name group_template_name
                           in Object_information obj_info in
                           Deadline_Time deadline);
  void destroy_subgroup(in DN distinguished_name);
  void enable_subgroup(in DN distinguished_name);
  void disable_subgroup(in DN distinguished_name);
  // 보안 서비스 //
  boolean request_ACL(in DN distinguished_name);
};
```

(그림 5) 그룹관리자 객체의 IDL

```
interface Timer {
  // 타이머 객체의 속성 //
  attribute TimeStruct   global_time;
  // 클라이언트의 시간 설정 //
  boolean Settime(in TimeStruct client_time)
  // 클라이언트의 시간 습득 //
  TimeStruct Gettime(in TimeStruct tm);
  // 시간값을 보여주기 위한 오퍼레이션 //
  boolean DisplayTime(in TimeStruct tm);
};
```

(그림 6) 타이머 객체의 IDL

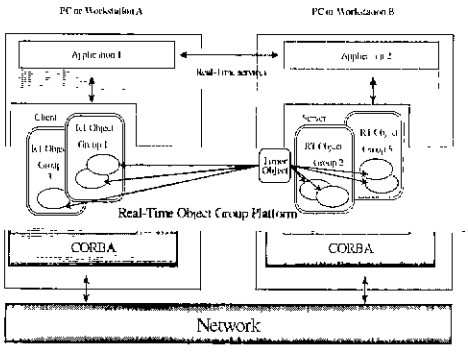
```
interface Scheduler {
  // 스케줄러 객체의 속성 //
  attribute short              schIdx;
  attribute short              timeInfoIdx;
  // 객체에 대한 스케줄 정보 습득 //
  boolean SettimeInfo(in string obj_name,
      in TimeSt exec_time, in TimeSt sync_time);
  // 객체에 대한 시간 정보 통보 //
  TimeInfoList GettimeInfo(in string obj_name);
  // 스케줄링 서비스 //
  boolean Scheduling(in string obj_name in TimeSt dead_line,
                     in string svc_name);
  // 서비스 객체의 우선순위 정보 서비스 //
  boolean Check_priority(in string obj_name);
  boolean EraseList(in string obj_name);
};
```

(그림 7) 스케줄러 객체의 IDL

4. 실시간 객체그룹 플랫폼의 구현

4.1 분산 환경

실시간 객체그룹은 CORBA를 기반으로 하며, 또 다른 플랫폼과 같은 역할을 한다. 실시간 객체그룹은 물리적인 이더넷이나 ATM 망을 기반으로 CORBA 환경에

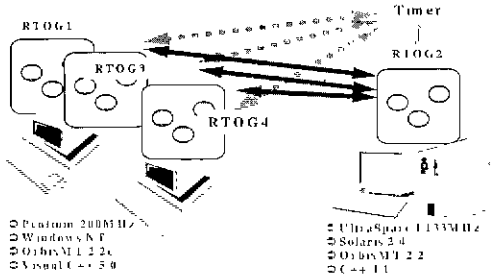


(그림 8) 실시간 객체그룹 플랫폼 지원 분산환경

서 구성되며, 실시간 어플리케이션 개발자는 실시간 객체그룹 모듈을 이용하여 원하는 어플리케이션을 구현할 수 있다. 실시간 객체그룹 플랫폼을 기반으로 개발한 어플리케이션에서 객체그룹간의 호출은 CORBA의 ORB를 통하여 이루어진다. (그림 8)은 본 연구가 이루어지는 실시간 객체그룹 지원 분산환경을 나타낸다.

4.2 개발 환경

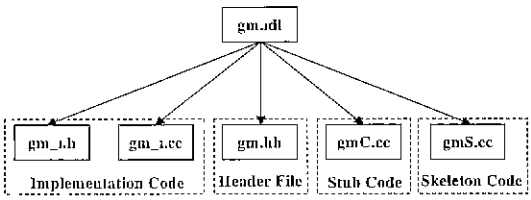
본 연구의 개발 환경은 클라이언트 측인 RTOG1, RTOG3, RTOG4는 Pentium 시스템에 운영체제는 WindowsNT 4.0을 사용하였으며, 미들웨어는 IONA Technologies의 Orbix MT 2.2c를 사용하였다. 구현코드 및 어플리케이션을 컴파일하기 위해서 Microsoft 사의 Visual C++ 5.0를 사용하였다. 서버 측인 RTOG2와 타이머는 SUN Microsystems의 UltraSparc-I 시스템으로 운영체제는 Solaris 2.4를 사용하였으며, 미들웨어는 IONA Technologies의 OrbixMT 2.2를 사용하였고, 구현코드를 컴파일하기 위해서 C++ 4.1을 사용하였다. (그림 9)는 본 연구의 개발환경이다.



(그림 9) 개발 환경

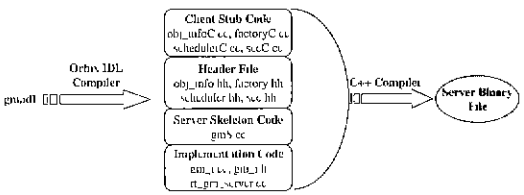
4.3 구현 과정

본 절에서는 실시간 지원 객체그룹 모듈을 구현하는 과정을 설명한다. 실시간 객체그룹에 대한 요구사항을 분석하고, 구성요소들을 정의하여 James Rumbaugh의 모델링 방법에 의해서 구성요소들의 기능을 객체, 기능, 동적 모델로 설계한다. 구성요소들 즉, 그룹관리자(gm.idl), 보안객체(sec.idl), ACL 객체(acl.idl), 스케쥴러(scheduler.idl), 객체팩토리(factory.idl), 객체정보 레포지토리(obj_info.idl), 타이머(timer.idl)의 인터페이스를 OMG-IDL로 설계한다. 각 구성요소의 IDL을 Orbix IDL 컴파일러를 통하여 컴파일 한다. 일반적으로 IDL 컴파일 후에는 클라이언트 측의 스텁 코드와 헤더화일, 그리고 서버 측의 스켈레톤 코드가 생성된다. 그렇지만, IDL 컴파일 시 -B -S 옵션으로 스텁 코드, 헤더화일, 스켈레톤 코드와 구현 코드를 생성한다. (그림 10)는 객체그룹의 대표적인 구성요소인 그룹관리자를 IDL 컴파일 후 생성되는 파일들을 나타낸 것이다.



(그림 10) 그룹관리자의 IDL 컴파일 후 생성 파일

위의 과정을 통하여 구성요소들의 모듈들을 IDL 컴파일 한다. 이렇게 생성된 파일 중에서 구현코드 내에 실제적으로 실행될 코드를 추가하고, 서버객체의 코드를 추가한 후, 자신과 관계된 객체들의 스텁 코드와 헤더화일, 그리고, 자신의 스켈레톤 코드를 함께 C++ 컴파일러를 사용하여 컴파일 한다. 이렇게 하여 서버 실행화일을 생성하게 된다. (그림 11)은 대표적으로 그룹관리자의 서버 실행 파일 생성과정을 나타낸 것이다.

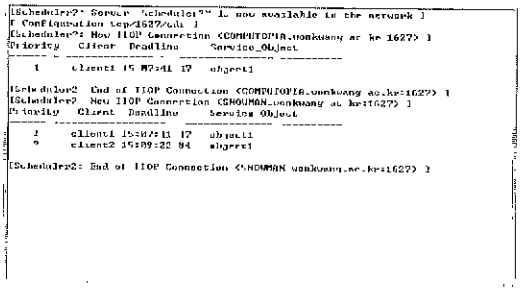


(그림 11) 그룹관리자의 실행파일 생성과정

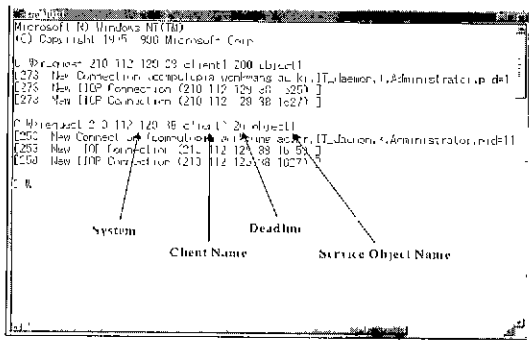
4.4 구성요소들의 실행화면

본 절에서는, 앞 절의 개발환경에서 구현된 실시간 객체그룹의 각 구성 요소들 중에서 마감시간을 갖는 클라이언트들이 다른 객체그룹 내에 서비스를 요청할 경우에 스케쥴러 내부에서 짧은 마감시간으로 스케쥴링하여 우선 순위가 역전(Inversion)된 스케쥴러의 실행화면을 차례로 나타낸다. (그림 12)은 스케쥴러에 클라이언트1과 클라이언트2의 요구 시에 마감시간에 의한 스케쥴링을 실행한 스케쥴러의 실행화면이다. (그림 13)는 클라이언트에서 객체그룹에 마감시간이 짧은 서비스를

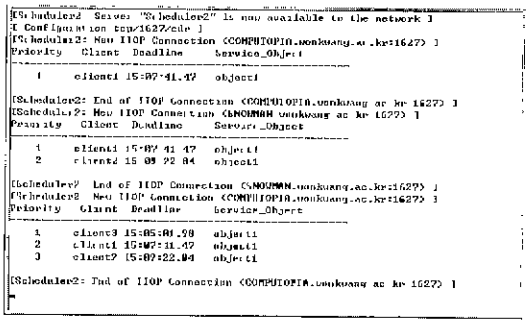
요구할 때의 요청화면이다. (그림 14)은 EDF(Earlist Deadline First)스케줄링에 의해 우선 순위가 역전된 실행화면이다



(그림 12) 스케줄러의 실행화면



(그림 13) 마감시간이 짧은 클라이언트의 요청화면



(그림 14) EDF 스케줄링에 의한 실행화면

5. 결 론

분산 컴퓨팅 환경은 통신망 관리의 복잡성을 해소하고 어플리케이션 개발자가 분산 이플리케이션을 설계할 때의 어려움을 해결하기 위해 분산 환경에서 객체를 설계하고 구현하는데 TINA, CORBA등의 표준화 방법을 사용하고 있다 이러한 분산 환경 하에서 복잡한 분산 소프트웨어의 개발 및 관리의 복잡성을 줄이고 분산된 객체들을 효율적으로 관리하면서 실시간 서비스를 위한 객체들의 접속 기능을 제공하기 위해 객체들의 모임인 실시간 지원 객체그룹의 정의가 필요하다. TINA에서 이미 객체그룹에 대한 정의가 제안되었으나, 객체그룹에 대한 구조나 구성요소들의 상세한 기능정립이 아닌 단순한 정의에 대한 기술이다 따라서, 본 논문은 TINA에서 제안한 객체그룹의 정의를 CORBA 기반으로 도입하여 객체그룹의 요구사항과 객체그룹내의 구성요소들에 대한 기능을 정립하고 실시간 지원 객체그룹의 모델 즉, 구조를 제시하였다. 또한, 실시간 서비스 수행을 위해 시간 제약조건을 만족하는 구성요소(스케줄러, 타이머)를 추가하여 실시간 객체그룹 플랫폼을 구현하였다. 본 연구의 실시간 객체그룹 플랫폼은 특정한 운영체제(실시간 운영체제)나 이종의 분산 환경에 영향을 받지 않고, ORB 구조와 실시간 이플리케이션들의 중간에 위치하여 실시간을 필요로 하는 분산된 객체들의 효율적인 관리와 이플리케이션의 개발을 용이하도록 하였다.

앞으로의 연구로는 멀티미디어 서비스를 위한 실시간 객체그룹 플랫폼으로 구조를 확장하고 실시간으로 서비스할 수 있는 멀티미디어 어플리케이션으로 응용하여야 할 것이다

참 고 문 헌

[1] OMG. "The Common Object Request Broker: Architecture and Specification revision 2.2," http://www.omg.org/corba/corbaCB.htm, 1998.

[2] Nguyen Duy Hoa, "Distributed Object Computing with TINA and CORBA. Technical Report Nr 97/7, http://nenya.ms.mff.cuni.cz/thegroup/. 1997.

[3] OMG Realtime Platform SIG, "Realtime CORBA A White Paper - Issue 1.0" http://www.omg.org/realtime/real-time_whitepapers.html, 1996.

[4] Victor Fay Wolfe et al., "Expressing and Enforcing Timing Constraints in a Dynamic Real-Time CORBA System," http://www.cs.uri.edu/rtsorac/publication.html. 1997

[5] Victor Fay Wolfe, et al, "Real-Time CORBA," In proceedings of the third IEEE Real-time Technology and Applications Symposium, 1997.

[6] James Rumbaugh, "Object-Oriented Modeling and Design," Prentice Hall, 1991.

[7] OMG, "Realtime CORBA 1.0 RFP," OMG Document: orbos/97-09-31, 1997.

[8] OMG TC, "Realtime CORBA Extensions - Joint Initial Submission," OMG TC, Document orbos/98-01-09, 1998.

[9] OMG, "CORBA Services : Common Object Services Specification," http://www.omg org/corba/sectran1 htm, 1997.

[10] "Scheduling in Real-Time Systems," http://www realtime-os.com/sched_o3.html, 1996.

[11] Andrew S. Tanenbaum, "Distributed Operating Systems," Prentice Hall International Inc 1995.

[12] G. P. A. Fernandes and I A. Utting, 'An Architecture for Scheduling of Services in a Distributed System," 1996.

[13] 주수종, 한국전자통신연구원, "분산처리환경에서 객체그룹 모델링 및 성능분석에 관한 연구", 최종보고서, 1997.

[14] 고창록, 신영석, 김명희, 주수종, "개방형 분산시스템에서 객체그룹 모델링에 관한 연구", 한국정보과학회 학술지, Vol.24, No.2, 1997.

[15] 신경민, 김명희, 주수종, "CORBA 환경에서 실시간 서비스 지원을 위한 분산 객체의 그룹화 및 관리", 한국정보처리학회, 제6권 5호, 1999.

[16] 김명희, 주수종, "실시간 태스크의 마감시간 만족을 위한 캐쉬 최적 분할 형태의 분석", 한국정보처리학회, 제4권 11호, 1997

김 명 희

e-mail : hee@wonkwang.ac.kr

1993년 원광대학교 컴퓨터공학과 공학사

1996년 원광대학교 컴퓨터공학과 공학석사

1996년~현재 원광대학교 컴퓨터공학과 박사과정 중

관심분야 : 분산 실시간 컴퓨팅, 시스템 최적화, 분산운영체제

이 재 완

e-mail : jwlee@ks.kug.ac.kr

1984년 중앙대학교 전자계산공학과 공학사

1987년 중앙대학교 컴퓨터공학과 공학석사

1992년 중앙대학교 컴퓨터공학과 공학박사

1997년~한국학술진흥재단부설 첨단학술 정보센타 전문위원 겸직

1992년~현재 군산대학교 전자정보공학부 교수

관심분야 : 분산 실시간 컴퓨팅, 분산객체모델, 운영체제

주 수 종

e-mail : scjoo@wonkwang.ac.kr

1986년 원광대학교 전자계산공학과 공학사

1988년 중앙대학교 컴퓨터공학과 공학석사

1992년 중앙대학교 컴퓨터공학과 공학박사

1993년 미국 Univ of Massachusetts at Amherst 전기 및 컴퓨터공학과 Post-Doc

1990년~현재 원광대학교 컴퓨터공학과 교수

관심분야 : 분산 실시간 컴퓨팅, 분산객체모델, 시스템 최적화, 멀티미디어 데이터베이스, 전자도서관, 분산운영체제