

분산 객체들의 관리와 실시간 서비스를 위한 플랫폼의 연구

김명희*, 주수중*

*원광대학교 전기전자 및 정보공학부

e-mail : hee@wonkwang.ac.kr, scjoo@wonkwang.ac.kr

A Study of Platform for Management and Real-Time Service of Distributed Objects

Myung-Hee Kim*, Su-Chong Joo*

*Division of Electricity Electronic and Information, WonKwang University

요 약

최근의 컴퓨팅 환경은 이질적인 클라이언트와 서버들간의 상호 운용성을 요구하는 분산 어플리케이션을 위한 프로그래밍 패러다임을 지원하기 위한 분산 객체 컴퓨팅 환경으로 발전되고 있다. 여기에는 복잡한 네트워크와 다양한 멀티미디어 응용 서비스를 위한 객체지향 기술들이 접목되고 있다. 이러한 분산 컴퓨팅 환경에서 처리되는 어플리케이션들의 실시간 서비스 지원을 위해 요구되는 실시간 특성과 분산 객체들의 관리의 어려움을 해결하기 위하여 본 논문에서는 실시간 객체그룹 플랫폼을 구축한다. 기존의 연구들은 실시간 CORBA를 사용하거나 또는 ORB를 수정하거나 실시간 운영체제상에 특정 CORBA 제품군을 사용하여 분산 환경의 특정 영역의 성능향상만을 도모하고 있다. 그러므로, 본 논문은 ORB의 수정 없이 표준 CORBA상에서 실시간 특성을 지원할 수 있는 실시간 객체그룹 플랫폼을 설계한다. 본 논문의 실시간 객체그룹 구조는 객체들의 관리적인 측면과 실시간 어플리케이션 서비스 지원 측면에 대한 요구사항들을 분석하여 정립한 모델이며, 구성요소들의 기능을 관리와 서비스로 분리하여 각각의 기능수행시에 발생하는 객체간의 상호작용이 다른 기능에 영향을 미치지 않도록 한다. 또한, 구축된 플랫폼은 실시간 어플리케이션 개발자에게 실시간 특성 파라미터의 표현과 처리에 대한 투명성을 부여하여 어플리케이션에 유연성과 확장성을 제공하기가 용이하도록 한다. 따라서, 본 논문은 실시간 객체그룹 플랫폼의 구성요소들에 대한 역할을 정의하고 기능을 정립하며, 각 구성요소들을 설계하고 구현하였으며, 구현되어진 실시간 객체그룹 플랫폼의 기능과 수행 성능을 검증한다.

1. 서론

분산 객체 컴퓨팅은 이기종의 클라이언트와 서버간의 상호운용성을 요구하는 어플리케이션을 위한 프로그래밍 패러다임이 되어가고 있다. OMG는 그러한 분산 환경을 위한 표준 소프트웨어 규격으로 CORBA(Common Object Request Broker Architecture)[1,3]를 개발하였다. 이 표준은 IDL(Interface Definition Language)을 이용하여 분산 컴포넌트(component)들의 기능적 행동(functional behavior)에 대한 인터페이스를 기술하도록 하고, 객체 서비스(즉, Naming, Event 등)와 분산된 클라이언트 객체와 서버 객체간의 상호작용을 고려한 미들웨어인 ORB (Object Request Broker)등을 기술하고 있다.

공장 자동화 제어, 항공분야와 같은 많은 분산 실시간 어플리케이션은 CORBA와 같은 분산 아키텍처로부터 이득을 얻을 수 있다. 많은 이들 어플리케이션의 설계자들은 그들의 아키텍처를 위해 CORBA를 고려했다. 그러나 현재 표준 CORBA는 실시간 요구사항을 지원하기가 부적절하다. 예를 들면, 분산 컴포넌트들의 기능적 행동에 대한 인터페이스를 기술하는 IDL을 이용하여 그들의 행동에 대

한 시간제약조건을 명확하게 기술할 수가 없다. 더 나아가, 분산 환경에 의해 제공되는 시스템 서비스들은 환경 전반에 걸친 중단간 실시간 스케줄링이나 클럭 동기화와 같은 기본적인 서비스들도 제공하지 않고 있다.

최근에, Real-Time SIG(Special Interest group)[1]가 CORBA 표준을 검사와 실시간 어플리케이션을 지원하기 위한 요구사항들을 정의할 목적으로 OMG내에 설립되어졌다. 특히, 실시간 SIG는 현재 CORBA 표준을 확장(CORBA/RT)함으로써 시간제약조건을 표현하고 실행할 수 있는 능력을 지원하는데 초점을 맞추고 있다. Real-Time SIG는 실시간을 지원하기 위한 분산 객체 컴퓨팅 환경을 위해 요구되었던 사항들을 자세히 다루는 백서(white paper)를 내놓았다. 이에 따라 세계적으로 많은 연구들이 이루어지고 있으나, 여기에는 CORBA의 핵심이 되는 ORB를 수정하거나 새롭게 확장하여 실시간을 지원하도록 하는 실시간 CORBA[5]를 제시하고 있다. 본 논문은 기존의 표준 CORBA[3]상에서 ORB를 수정하거나 실시간 운영체제[11]위에 포팅(porting)시키지 않고, 실시간을 지원 가능하게 할 수 있는 구조에 초점을 맞춘다. 또한, 분산된 객체들의 효율적이고 체계적인 관리와 실시간 지원을 위해 개별 서비스 객체들에게 과중 되는 관리 절

*본 연구는 2001년 정통부 대학기초 연구지원사업(2001-007-3)의 지원으로 수행되었음.

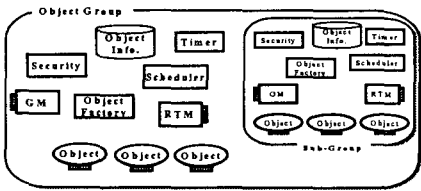
차를 간편하게 하기 위해 기존에 연구가 되어진 객체그룹의 개념[6,7]을 도입한다. 이렇게 함으로써, 본 연구는 기존의 표준 CORBA에서 분산 객체들의 관리와 분산 실시간 특성을 지원할 수 있는 실시간 객체그룹 모델을 제시하며, 이에 따른 요구사항과 기능들을 정립하며, 관리와 서비스측면에서 객체들간의 상호작용의 절차를 기술한다. 또한, 실시간 객체그룹 플랫폼에 대한 성능분석을 통해 본 논문의 추후 연구 과제와 결론을 제시한다.

2. 실시간 객체그룹 모델

실시간 시스템에서, 실시간 특성을 지원하기 위해서는 여러 가지 특정 환경 하에서 시간제약조건을 만족시켜야 하기 때문에, 이를 분산 객체 컴퓨팅 환경과 접목시키기는 쉽지 않다. 현재 실시간 운영체제에 CORBA를 포팅시키거나 Fast CORBA와 같은 분산 환경에 실시간 특성을 접근시키는 연구가 이루어지고 있으나, ORB를 수정하거나 실시간 특성의 완전한 지원이 되지 않는 문제점들을 가지고 있다. 따라서, 본 논문은 분산 객체 컴퓨팅 환경으로 표준 CORBA를 기반으로 하며, 객체들의 관리를 용이하게 하기 위한 객체그룹 구조에 현재 컴퓨팅 환경에서 요구되고 있는 중단간 서비스를 실행하는데 실시간 조건을 만족하는 실시간 객체그룹(Real-Time Object Group, 이하 RTOG) 플랫폼을 제공하고자 한다.

2.1 실시간 객체그룹의 구조

RTOG의 객체는 실시간 서비스를 수행하기 위해, 실시간 특성을 지원한다. 다시 말해서, 객체에 대한 서비스 요청은 단순한 메소드 요청이 아닌, 마감시간과 같은 시간제약조건들을 포함한 메소드 요청으로 이루어진다. 앞서 말한바와 같이 기존의 연구에서는 이러한 시간제약조건을 표준 CORBA상에 ORB의 수정없이 분산된 서비스 객체에 전달하는 것이 불가능하므로, 이를 가능토록하는 RTOG 플랫폼을 제시하는데 초점을 맞춘다. (그림 1)은 RTOG의 구조이다.



(그림 1) RTOG의 구조

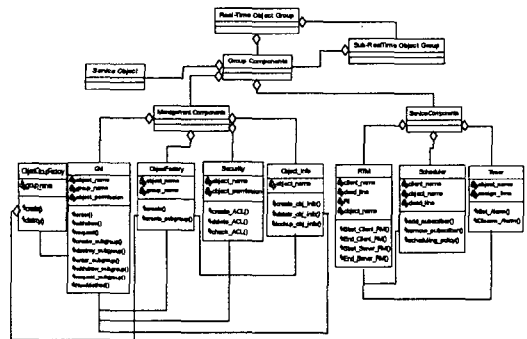
RTOG는 그룹관리자(Group Manager:GM), 실시간관리자(Real-Time Manager : RTM), 객체 팩토리(Object Factory), 보안 객체(Security), 스케줄러(Scheduler), 타이머(Timer), 객체정보 레포지토리(Object Information Repository : Object Info.), 서비스 객체(Service Object)들과 서브객체그룹(Sub-Group)으로 구성된다.

RTOG의 구성 요소들은 분산 객체의 관리적 기능 수행 객체들과 서비스 호출 수행 객체들로 나눌수 있다. 먼저, 관리적 기능 수행객체들로 그룹관리자는 객체그룹에 대한 관리적인 책임을 지며, 모든 객체그룹내 서비스 객체들에 대한 서비스 요청에 필요한 관리적인 절차를 객체

그룹내의 다른 구성 요소 객체들 즉, 보안 객체, 객체 정보 레포지토리, 객체 팩토리과 함께 상호작용하며 수행한다. 객체 팩토리는 그룹관리자의 객체 생성 요구에 따라, 객체 정보 레포지토리의 정보를 이용하여 서비스 객체를 생성한 후, 생성된 객체의 레퍼런스를 그룹관리자에게 반환하며, 서비스 객체와 서브객체그룹을 하나의 객체로 취급한다. 보안객체는 객체그룹 외부의 모든 클라이언트 요청에 대해, 접근 제어 리스트(Access Control List:ACL)와 접근 규칙을 참조하여 접근 권한 검사에 대한 모든 절차를 수행한다. 객체 정보 레포지토리는 객체그룹내의 서비스 객체들과 서브객체그룹의 그룹관리자 객체에 대한 생성과 관련된 정보를 관리한다.

실시간 서비스 수행에 관련된 객체들은 실시간 관리자, 타이머 객체, 스케줄러이다. 실시간 관리자는 분산 객체들에게 시간제약조건의 처리에 대한 투명성을 제공하며, 서비스 객체에게 실시간 특성을 부가하여, 이에 따른 처리를 위해 스케줄러, 타이머와 상호 작용한다. 실시간 관리자는 서비스를 요청하는 객체(클라이언트)가 소속되어 있는 그룹 내에 있는 실시간 관리자와 서비스 객체가 소속되어 있는 그룹 내에 있는 실시간 관리자의 기능이 구별된다. 타이머는 클라이언트나 서비스 객체에게 요청된 마감시간의 알람기능을 제공하고, 스케줄러는 특정 실시간 스케줄링 알고리즘에 의해 구현된 객체이다. 스케줄러는 적용되는 스케줄링 알고리즘에 따라 클라이언트들의 우선순위를 선정하여, 서비스 객체들이 이 우선순위에 따라 서비스를 수행하도록 한다. 스케줄러는 서비스 요청이 들어온 서비스 객체들마다의 스케줄링 리스트(우선순위 리스트)를 간직하여 우선순위를 스케줄링한다.

RTOG의 구성 요소 객체들에 대한 전체적인 클래스 다이어그램은 (그림 2)에 나타난다.



(그림 2) RTOG의 클래스 다이어그램

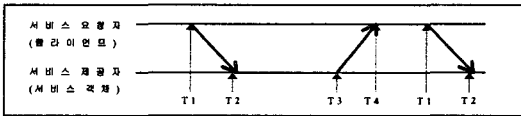
2.2 시간제약조건

분산 처리 환경 하에서 클라이언트가 서비스를 요청하여 결과를 반환 받을때까지의 기본 흐름은 서비스 요구단계, 서비스 처리 단계, 결과 전달 단계 등 크게 3단계로 구분된다. 이들 각 단계마다 시간적인 제약조건이 있으며, 이를 명확하게 정의하여 적시성(timeliness)을 보장해야 한다. 이를 위한 시간제약조건들은 다음과 같이 크게 5가지로 정의한다.

- ▶ 호출시간(IT : Invocation Time)
 - 클라이언트의 호출 (CIT), 서비스객체의 호출 (SIT)

- ▶ 수행시간 (ST : Service Time)
- ▶ 전송시간 (TT : Transfer Time)
- ▶ 서비스 수행 마감시간 (SD : Service Deadline)
- ▶ 서비스 요청 마감시간 (RD : Request Deadline)

IT는 호출시간으로, CIT는 클라이언트가 서비스 객체에 서비스 요청하는 시간이고 SIT는 서비스 객체에 클라이언트의 요청이 도착한 시간 즉, 서비스 객체의 실제 호출시간이다. ST는 서비스 객체가 서비스를 수행하는데 걸리는 시간이며, TT는 클라이언트의 요청이 서비스 객체에 도착하는 데까지의 전송시간 즉, ST-IT를 의미한다. SD는 서비스 객체가 반드시 완료되어야 하는 시점이며, RD는 클라이언트가 서비스를 요구한 후에 수행 결과가 반환되어야 하는 시점을 의미한다. 여기에서 RD가 분산 실시간 처리 시스템에서 보장되어야 하는 제약조건이다. (그림 3)는 이들간의 관계를 보여준다.



- T1 : 클라이언트의 한 서비스 요청에 대한 호출시간 (CIT)
- T2 : 서비스 객체의 호출시간 (SIT)
- T3 : 서비스 객체의 서비스 수행 마감시간 (SD)
- T4 : 클라이언트의 서비스 요청에 대한 마감시간 (RD)

(그림 3) 시간제약조건의 정의

이 정의를 통하여 시간제약조건을 다시 표현하면 다음과 같다.

$$TT = SIT - CIT, ST = SD - SIT$$

$$RD > ST + (2 \times TT) + slacktime$$

$$SD < (RD - TT)$$

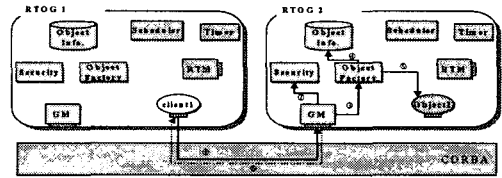
서비스 실행 데드라인(SD)은 서비스 실행시간 ST와 전송시간(TT) 그리고 slack time을 합한값이다. 이 데드라인은 실시간 서비스를 보장하기 위하여 클라이언트의 요청 데드라인(RD)에서 전송시간을 뺀시간보다는 작아야 한다.

3. RTOG 구성요소들의 상호작용

이 장에서는, RTOG에서 처리하는 요청들에 대한 관리적인 수행 절차와 실시간 서비스 수행에 따른 절차를 접속 절차도로 나타낸다.

3.1 객체 관리를 위한 수행 절차

RTOG의 모든 관리에 대한 요청은 그룹관리자를 통하여 일어나게 되며, 요청할 수 있는 관리적 기능에는 분산되어 있는 서비스 수행객체를 RTOG로 포함, RTOG에 소속되어 있는 객체의 탈퇴, 클라이언트의 서비스 객체에 대한 레퍼런스 요청, RTOG내 서브객체그룹의 생성, 서브객체그룹으로의 서비스 객체의 포함, 탈퇴, 서브객체그룹내의 서비스 객체에 대한 레퍼런스 요청 등이 이에 해당된다. 이러한 관리적 기능 수행절차들 중에, 클라이언트의 서비스 객체 레퍼런스 요청에 대한 객체간의 상호 접속 절차를 예로 들어 설명하며, (그림 4)는 이들간의 상호 접속 과정을 나타낸다.



(그림 4) 클라이언트의 레퍼런스 요청에 대한 접속 절차도

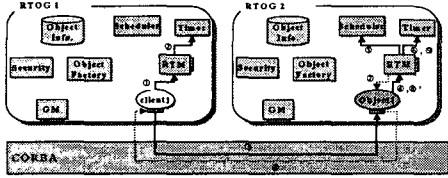
클라이언트는 서비스 객체가 포함되어 있는 객체그룹의 그룹관리자에 대한 레퍼런스를 네임서버를 이용하여 획득한 후, 이를 통하여 그룹관리자에게 서비스 객체의 레퍼런스를 요청하게 된다. 그룹관리자는 클라이언트가 요청한 서비스 객체에 대한 접근 권한 검사를 보안 객체에게 넘기고, 보안 객체는 접근 권한 검사 수행 후 결과를 반환한다. 그룹관리자는 반환 받은 결과가 클라이언트의 접근 가능이면, 객체 팩토리에게 서비스 객체의 생성을 요청한다. 객체 팩토리는 생성을 요청 받은 서비스 객체의 정보를 객체 정보 레포지토리에서 참조하여 객체를 생성한 후, 생성된 객체의 레퍼런스를 그룹관리자에게 반환한다. 그룹관리자는 객체 팩토리 객체에게서 반환 받은 서비스 객체의 레퍼런스를 클라이언트에게 반환한다.

3.2 실시간 서비스를 위한 수행 절차

클라이언트의 서비스 수행 요청은 먼저 관리 절차에 따라 서비스 객체의 레퍼런스를 획득한 후에 요청된다. 이때 앞서 말한바와 같이, 실시간 특성을 가진 서비스 요청이므로 클라이언트의 오퍼레이션 호출에는 반드시 시간제약조건을 나타내는 파라미터(Real-time Information, RI)가 포함되어져야 한다. 또한, 각 RTOG의 실시간 관리자는 전역 시간 서비스를 이용하여 분산 환경에서의 전역 시간 정보를 계속 유지하고 있다고 전제한다.

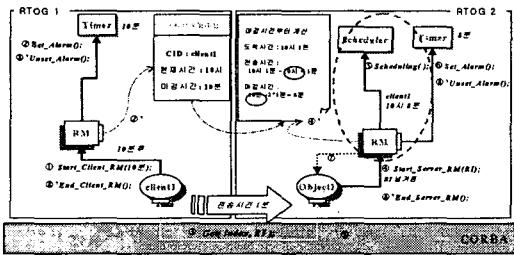
클라이언트는 다른 객체그룹에 있는 서비스 객체의 오퍼레이션을 호출하기 전에, 자신의 객체그룹내에 있는 실시간 관리자에게 마감시간 정보를 넘겨준다. 실시간 관리자는 현재 전역 시간과 클라이언트가 요구한 마감시간에 대한 상대값을 RI에 저장한다. 실시간 관리자는 클라이언트가 준 마감시간에 따라 자신의 객체그룹에 있는 타이머를 설정하며, 이 타이머는 클라이언트에게 마감시간의 종료를 인식시키고, 클라이언트가 그에 따른 처리를 수행하게 한다. 클라이언트의 객체그룹에서 이러한 일련의 과정이 수행된 후, 클라이언트는 다른 객체그룹에 있는 서비스 객체에 실질적인 서비스를 요청하며, 이때 앞서 수행된 결과를 저장하고 있는 RI도 같이 서비스 객체에게 넘겨진다. 서비스 객체는 넘겨받은 RI를 자신의 객체그룹에 있는 실시간 관리자에게 전달하고, 실시간 관리자는 RI를 이용하여, 전송시간을 고려한 서비스 수행에 필요한 마감시간을 계산한 후, 계산된 마감시간을 스케줄러에게 넘겨주어, 클라이언트의 서비스 수행 우선순위를 스케줄링 하게 한다. 스케줄링의 우선순위에 따라 타이머를 해당 클라이언트의 계산된 마감시간으로 설정하며, 이 시간값은 서비스 객체를 포함하는 실시간 관리자에게 마감시간의 종료를 확인시키며, 이에 따른 처리를 수행토록 한다. 실시간 관리자가 서비스 객체에게 우선순위에 따른 클라이언트의 정보를 반환하면, 객체는 서비스를 수행한 후, 해당 클라

이언트에게 결과값을 반환하고 실시간 관리자에게 서비스 수행이 끝났음을 알린다. 실시간 관리자는 타이머의 설정을 해제하고, 결과값을 반환 받은 클라이언트도 자신의 객체그룹에 있는 타이머의 설정을 해제함으로써, 하나의 서비스 수행에 따른 각 객체들의 처리절차를 마감하게 된다. (그림 5)는 실시간 서비스 수행에 따른 객체간의 접속 절차를 나타낸다.



(그림 5) 실시간 서비스 요청에 대한 접속 절차도

(그림 6)은 RTOG의 구성요소들이 실시간 서비스를 수행하면서, RI 파라미터에 시간제약조건들을 어떻게 표현하고 처리하는지를 예를 들어 설명한다.

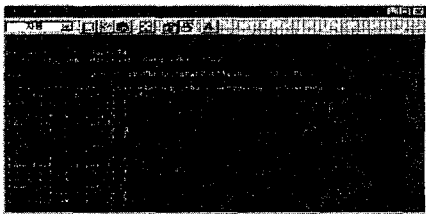


(그림 6) RI 파라미터 사용에 대한 절차도

4. 구현과 검증

RTOG 플랫폼은 이기종의 분산 컴퓨팅 환경을 독립적으로 지원하기 위하여 IDL로 설계되고, Window98상에서 동작하는 비즈브로커와 비주얼 C++로 구현되었다.

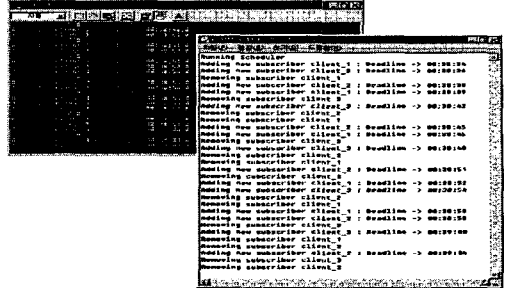
구현된 RTOG 플랫폼의 수행 절차를 검증하기 위하여, 본 논문에서는 클라이언트들의 실시간 서비스 요청에 따른 스케줄러 객체와 실시간 관리자 객체의 수행을 보여준다. (그림 7)은 실시간 관리자의 수행화면을 나타내며, 각 클라이언트들의 서비스 요청에 대한 수행 성공을 나타낸다. 이를 결과는 주어진 클라이언트의 데드라인과 스케줄링 정책에 따라 다르게 나타날 수 있다.



(그림 7) 실시간 관리자의 수행 화면

(그림 8)은 클라이언트의 우선순위가 스케줄러의 스케줄링 정책의 기준에 따라 변화됨을 볼 수 있다. 본 실험절차에서 스케줄링 정책으로 EDF 스케줄링 알고리즘을 적용시켰기 때문에, 마감시간이 짧은 클라이언트의 서비스 요청

의 우선순위가 높게 된다. 또한, 스케줄링 리스트로부터 클라이언트의 요청이 삭제되는 두가지 경우를 볼 수 있는데, 서비스 수행의 마감시간이 위배된 경우와 아직 마감시간이 위배되지 않았지만 남아있는 시간이 현재 서비스 수행에 걸리는 시간보다 적게 남아있는 경우이다.



(그림 8) 스케줄러의 수행 화면

5. 결론

본 논문은 실시간 CORBA나 실시간 운영체제 상에서 분산 객체 컴퓨팅에 실시간 특성을 접목시키지 않고, 표준 CORBA 환경에 객체간의 메소드 호출에 대한 수행 시에 실시간 특성을 지원할 수 있는 방향으로 연구를 하였다. 또한, 분산 객체들의 관리의 용이성을 제공하기 위하여 객체그룹의 개념을 접목시켜 새로운 객체그룹 모델을 제시하고 있다. 이에 따른 실시간 객체그룹 플랫폼에 대한 설계와 구현 그리고 구성요소들에 대한 수행 검증을 위한 시뮬레이션을 통해 본 실시간 객체그룹 플랫폼의 타당성과 기능 검증을 보여주었다.

실시간 객체그룹은 실시간 정보(RI) 구조체를 정의하여 실시간 파라미터를 처리하였고, 타이머를 두어 클라이언트와 서버 각각의 마감시간 확인을 용이토록 하였으며, 스케줄러를 따로 두어 플랫폼을 사용하는 시스템이나 어플리케이션의 상황에 맞는 스케줄링 정책을 유동적으로 사용할 수 있도록 하였다.

추후 연구로는 RI에 포함되는 파라미터의 종류들을 확장시키고, 스케줄러가 사용하는 스케줄링 정책을 클라이언트의 특성에 맞게 선택할 수 있는 구조로 보완하고자 한다.

참고문헌

[1] OMG Realtime Platform SIG, "Realtime CORBA A White Paper", http://www.omg.org/realtime/real-time_whitepaper.html
 [2] Victor Fay Wolfe, et al., "Real-Time CORBA", In proc. of the third IEEE Real-time Technology and Applications Symposium, 1997.
 [3] OMG, "The Common Object Request Broker: Architecture and Specification 2.2", <http://www.omg.org/corba/corbaCB.htm>, 1998.
 [4] John. K. Black, et al. "Real-time Method Invocations in Distributed Environments", University of Rhode Island Department of Computer Science and Statics, Technical Reports TR95-244, 1995.
 [5] OMG, "CORBA Services: Common Object Services Specification", <http://www.omg.org/corba/sectran1.htm>, 1997.
 [6] 신경민, 김명희, 주수중, "CORBA환경에서 실시간 서비스 지원을 위한 분산 객체의 그룹화 및 관리", 정보처리학회 논문지, 6권5호, 1999. 5
 [7] 김명희, 주수중, "CORBA환경에서 실시간 응용 지원을 위한 분산 객체그룹 플랫폼의 설계 및 구현", 정보처리학회 논문지, 7권4호, 2000. 4