분산 실시간 서비스를 위한 CORBA 객체그룹 플랫폼의 구축

(Construction of CORBA Object-Group Platform for Distributed Real-Time Service)

김 명 희 † 주 수 종 ††
(Myung-Hee Kim) (Su-Chong Joo)

요 약 최근의 컴퓨팅은 이질적인 클라이언트와 서버들간의 상호 운용성을 요구하는 분산 어플리케이션을 위한 프로그래밍 파라다임을 지원하기 위한 분산 객체 컴퓨팅 환경으로 발전되고 있다. 여기에는 복잡한 네트워킹과 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 기능 수행시에 발생하는 객체간의 상호작용이 다른 역할에 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.

Abstract Recently, the computing has developing in distributed object computing environment for supporting a programming paradigm of distributed application requiring interoperability between heterogeneous clients and servers. It involves the complex networking and the object-oriented technologies for various multimedia application service. In this paper, we construct the real-time object group platform for solving the difficulties of managements of distributed objects and the real-time constraints by requiring for real-time service supporting of applications in distributed computing environment. The existing researches are being tried to only improving the performnace of systems by using real-time CORBA itself, or modifying the part of CORBA compliance. Hence, we design a new model of real-time object group platform that can support the real-time requirement without modifying the ORB. The structure of our real-time object group analyzed and defined the requirement about object management and real-time application service sides. And the role of the components of real-time object group is divided into 2 classes for reducing the side effect of interoperability between management and service. Also, we considered how to transparently express the parameters of real-time properties for clients and developers of server's service objects. If the expression of real-time parameters is transparent, then the developer can easily extend the real-time parameters simply and flexibly. Therefore, in this paper we defined the role of components of platform and described functions of each component and designed and then implemented the real-time object group platform. Finally, we showed the execution procedures of implemented our platform for verifying the functionality.

· 이 논문은 2000년도 원광대학교의 교비지원에 의해서 연구됨
† 정 회 원 : 원광대학교 전기전자 · 정보공학부 교수
hee@wonkwang.ac.kr
†† 종신회원 : 원광대학교 컴퓨터 · 정보통신공학부 교수
scjoo@wonkwang.ac.kr
논문접수 : 2001년 3월 6일
심사완료 : 2001년 9월 26일

1. 서 론

분산 객체 컴퓨팅은 OSF의 DCE나 OMG의 CORBA와 같은 환경에서 발전되고 있다. 특히 CORBA[3]는 분산 환경에 대한 표준 소프트웨어 규정(specification)으로 개발되었다. CORBA는 IDL을 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. 최근에, 이러한 컴퓨팅 환경에 실시간 어플리케이션에 대한 요구가 증대되고 있다.

일반적인 실시간 시스템은 시간적인 제약 조건[1,4,5,7]들을 갖고 있다. 즉, 임의의 응용 시스템의 동작 중에 어떤 한 상태가 주어진 시간 안에 이루어지지 않는 경우에 치명적인 영향을 주는 경우가 대부분이다. 이와 같이 실시간 시스템의 주된 목적은 모든 응용 시스템들이 각자에게 주어진 시간적인 요구 조건을 정확히 만족할 수 있도록 보장하는 것이다. 최근 들어 이러한 실시간 처리를 요구하는 시스템들이 보다 더 증가하고 있다. 예를 들어, 지난 수 십 년 동안 실시간 성격을 가지는 통신 시스템과 군사 시스템을 비롯하여 최근에는 전자상거래를 포함한 멀티미디어 응용 시스템, 전자도서관 시스템 등의 클라이언트/서버 구조를 갖는 분산 환경의 응용 시스템들도 시간적인 제약 조건들을 내포하고 있다. 클라이언트/서버 형태의 분산 처리 시스템 내에서 클라이언트내의 응용 시스템들은 서버에게 특정 시간 안에 서비스가 지원될 수 있는 실시간 처리 시스템을 요구하고 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 부적절 하다. 예를 들면, IDL은 분산 컴포넌트들의 기능적 행동에 대한 인터페이스를 기술한다. 그러나 IDL을 이용하여 그들의 행동에 대한 시간 제약조건을 명확하게 기술할 수가 없다. 더 나아가, 분산 환경에 의해 제공되는 시스템 서비스들은 환경 전반에 걸친 종단간 실시간 스케쥴링[4,10,12]을 위한 지원을 거의 제공하고 있지 않다. 실제로 일부 분산 환경들은 클럭 동기화와 같은 기본적인 서비스들도 제공하지 않고 있다.

최근에, Real-Time SIG(Special Interest group)[1,7]가 CORBA 표준을 검사하고 실시간 어플리케이션을 지원하기 위한 요구사항들을 정의할 목적으로 OMG내에 설립되어졌다. 특히, 실시간 SIG는 현재 CORBA 표준을 확장(CORBA/RT)함으로써 시간 제약조건을 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 절차를 간편하게 하기 위해 기존에 연구가 되어진 객체그룹의 개념[14,15,16]을 도입한다. 이렇게 함으로써, 본 연구는 기존의 표준 CORBA에서 분산 객체들의 관리와 분산 실시간 특성을 지원할 수 있는 실시간 지원 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. 배경연구

다수의 사용자가 정보를 공유할 수 있는 분산 컴퓨팅과 객체지향 기술의 장점을 효과적으로 통합하여 주는 분산 객체 컴퓨팅은 개발자에게 어플리케이션 개발의 생산성 향상과 사용자에게 분산 환경에 투명하게 통합된 정보를 제공해준다.

현재 컴퓨터 사용자들은 생산성 있는 응용 프로그램과 파일이나 데이터베이스의 공유를 원하며, 다른 사용자들과 정보를 공유하고 통신하길 원한다. 이러한 다양한 요구사항을 만족시키는 프로그램을 작성하기 위해서는 다른 운영체제와 네트워크 환경 등 이종의 환경을 기반으로 하는 클라이언트/서버 프로그램을 개발해야 하며, 이를 쉽게 하기 위해 RPC(Remote Procedure Call)와 OLE(Object Linking & Embedding) 방식이 나왔다. 그러나 RPC는 네트워크와 상호작용하는 프로토콜을 프로그램 개발자가 작성하는데 많은 시간이 소요되며, OLE는 윈도우즈 환경에서의 객체간 호환이므로 이를

해결하는 공통구조라는 개념의 필요성이 대두되었고, CORBA와 TINA[2]등의 네트워크에 기반을 둔 분산 객체지향 시스템의 설계와 개발에 대한 표준이 제시되었다.

CORBA는 클라이언트가 임의의 객체나 비객체지향적 프로그램을 요구할 때, 클라이언트로부터 하나의 요구를 받아들이고 응답하는 과정을 균일하게 제공하자는 것이 기본 개념이다. 이를 위해 인터페이스 정의어인 IDL이 요구되었고, 클라이언트측과 구현객체측 사이의 객체간 처리 과정이 기술되어졌다. 또한 응용 가능한 오퍼레이션은 객체 서비스로 두어 객체의 상호작용성과 기능성을 확장시켰다.

TINA는 이질적인 네트워크를 기반으로 한 단일 분산 객체지향 환경을 정의하고 있다는 것은 CORBA와 유사한 개념이지만, 여러 가지 부분에서 CORBA와 차이점을 보인다. 본 논문에서는 객체를 다루는 기본개념으로 TINA에서 사용하는 객체그룹 단위 개념을 도입하여, 분산 객체들의 관리를 도모하고자 하였다. 그러나, TINA는 아직까지 구현 제품군이 나와있지 않기 때문에 분산 객체 컴퓨팅 환경의 대표로 자리잡고 있는 CORBA를 기반으로 객체그룹의 개념을 적용시켜본 것이다. 분산 객체의 관리적 용이성을 목적으로 논리적인 객체의 집합체를 구성하여 객체그룹을 정의하고, 관리에 필요한 여러 가지 기능을 수행하는 구성요소를 정의하여 객체그룹의 구조를 기존에 제시하였다[14,15].

현재 요구되어지는 컴퓨팅 환경의 요구사항 중의 하나는 실시간 특성의 지원이다. 그러나, 실시간 특성을 지원하기 위해서는 여러 가지 특정 환경 하에서 시간 제약조건등의 특성을 만족시켜야 하기 때문에, 이를 분산 객체 컴퓨팅 환경과 접목시키기는 쉽지 않다. 현재 분산 환경에 실시간 특성을 접근시키는 연구[4,5]가 이루어지고 있으나, 이는 특정 환경을 위해 만들어지거나 또는 기본 분산 환경의 수정이나 확장을 필요로 한다. 가장 대표적인 연구[4,5]로서, 실시간 운영체제상에 표준 CORBA를 포팅(porting)시키는 연구가 있으나, 이는 일단 특정 운영체제상에서 동작하며, 실시간 특징을 제한적으로 사용한다. 또 다른 연구로 "fast" CORBA가 있으나 이 연구 역시 CORBA의 이벤트 서비스의 성능 향상에 목적을 두고 이루어지는 연구이며, ORB의 수정과 실시간 특징을 기술할 수 없다는 문제점을 지니고 있다. CORBA와 실시간 시스템을 통합한 실시간 CORBA는 시스템 환경과 실시간 CORBA가 반드시 지원해야 할 기능들에 대해 기술해 놓은 기술서로서, 아직까지 완벽한 제품군이 출시되고 있지 않다.

따라서, 본 논문은 분산 객체 컴퓨팅 환경으로 표준 CORBA를 기반으로 객체들의 관리를 용이하게 하기 위한 객체그룹 구조에 현재 컴퓨팅 환경에서 요구되고 있는 종단간 서비스를 실행하는데 실시간 조건을 만족하는 실시간 객체그룹 플랫폼을 제시하고자 한다. 여기에서 고려되어야 할 실시간 개념은 시간관리, QoS, 스케쥴링, 동시성, 고장허용, 성능등이며, 실시간 기술로는 CPU스케쥴링과 디스패칭 기술, 실시간 통신 프로토콜, 실시간 시스템에서 결함허용 소프트웨어의 구현, QoS 관리, 클럭 동기화등이다. 본 논문에서는 RT/SIG에서 고려한 실시간 요구 사항들[1]의 세가지 영역 즉, 오퍼레이팅 환경을 위한 요구사항, ORB 아키텍쳐를 위한 요구사항, 객체 서비스[9]와 기능을 위한 요구사항중에서 시간 제약조건을 표현하고 지원하는 객체 서비스와 기능을 위한 요구사항에 중점을 두고 연구한다. 이들 요구사항은 분산 컴퓨팅 환경에 포함되어져야 하는 CORBA의 메소드 호출, 전역 시간 서비스와 실시간 스케쥴링과 우선순위 결정 방법등의 분석이 포함된다. 또한, 기존의 ORB를 수정하지 않고 실시간 메소드 호출이 가능하도록 시간 제약조건의 표현 형식에 대한 연구도 포함한다.

3. 실시간 객체그룹

본 장에서는 실시간 객체그룹의 구조와 CORBA 표준에 따른 객체간의 상호작용을 나타내고, 실시간 지원을 위한 시간 제약조건의 표현과 처리에 대한 정의와 스케쥴링 절차에 대해 나타낸다.

3.1 실시간 객체그룹의 구조

실시간 객체그룹의 객체는 실시간 서비스를 수행하기 위해, 실시간 특성을 가지고 있다. 다시 말해서, 객체에 대한 서비스 요청은 단순한 메소드 요청이 아닌, 시간 제약조건(예를들면, 마감시간)등을 포함한 메소드 요청으로 이루어진다. 앞서 말한바와 같이 기존의 연구에서는, 이러한 시간 제약조건을 표준 CORBA상에 ORB의 수정없이 분산된 서비스 객체에게 전달하는 것이 불가

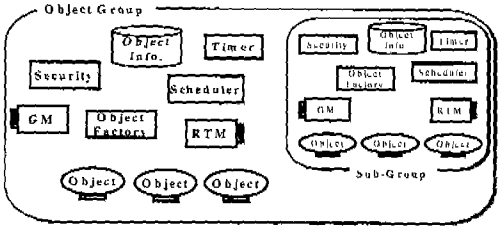


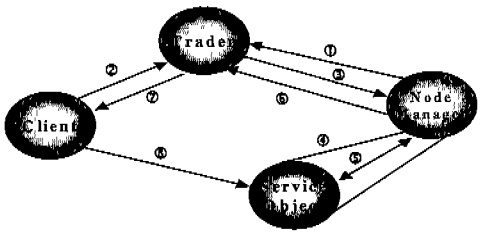
그림 1 실시간 객체그룹의 구조

능하므로, 이를 가능토록하는 실시간 객체그룹 모델을 제시하는데 초점을 맞춘다. (그림 1)은 실시간 객체그룹의 구조이다.

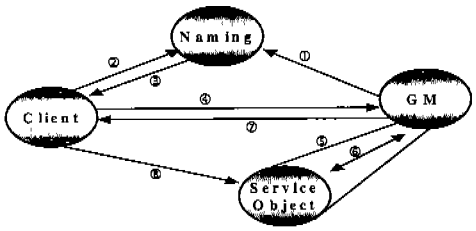
실시간 객체그룹은 그룹관리자(Group Manager : GM), 실시간관리자(Real-Time Manager : RM), 객체팩토리(Object Factory), 보안 객체(Security), 스케줄러(Scheduler), 타이머(Timer), 객체정보 레포지토리(Object Information Repository : Object Info.), 서비스 객체(Object)들과 서브객체그룹(Sub-Group)으로 구성된다.

3.2 그룹관리자의 기본적인 관리기능

실시간 객체그룹에서 그룹관리자는 그룹 내 객체들에 대한 관리적인 기능을 수행한다. 여기에는 분산 객체들을 특정 객체그룹에 소속시키거나 탈퇴시키는 기능과 서비스 요청을 위한 객체의 생성에 대한 기능을 수행한다. 기본적으로 관리기능을 수행하기 위해서는 클라이언트와 그룹관리자간 그리고 그룹관리자와 서비스 수행객체간의 상호 접속이 필요하다. 이는 일반적인 CORBA의 생명주기 서비스[9]와 유사하다. 따라서, 클라이언트가 서비스 수행객체에게 실제적인 서비스 요청을 하기까지의 기본적인 객체간의 접속 절차 과정을 CORBA의 생명주기 서비스에 연관시켜 설명한다. 물론, 접속 절차의 핵심 기능은 그룹관리자의 기능에 해당된다. 다음 (그림 2)의 (a)와 (b)는 각각 CORBA의 생명주기 서비스에 따른 객체간의 접속 절차와 실시간 객체그룹의 그룹관리자의 관리기능에 따른 객체간의 접속 절차를 나타낸다.



(a) CORBA의 경우



(b) 실시간 객체그룹의 경우

그림 2 객체간 접속 절차

CORBA의 생명주기 서비스는 트레이더 서비스를 사용하여 클라이언트가 서비스 요청을 할 객체의 레퍼런스 획득을 대행시키게 된다.

① 노드 관리자는 트레이더에 서비스 객체를 등록한다.
② 클라이언트는 원하는 서비스 객체를 요청한다.
③ 트레이더는 클라이언트가 요청한 객체를 등록한 노드 관리자에게 클라이언트의 요청을 전달한다.
④ 노드 관리자는 팩토리를 이용하여 서비스 객체를 생성한다.
⑤ 노드 관리자는 생성된 서비스 객체의 인터페이스를 통하여 초기화 오퍼레이션을 호출하여 객체의 레퍼런스를 반환 받는다.
⑥ 노드 관리자는 트레이더에 서비스 객체의 인터페이스를 반환한다.
⑦ 트레이더는 원래의 클라이언트에게 이 인터페이스를 반환한다.
⑧ 클라이언트는 서버에 오퍼레이션을 지금 호출할 수 있다.

실시간 객체그룹에서 지원되는 객체간 접속에는 우선 CORBA 상용제품들에서 기본적으로 지원되는 네이밍 서비스를 이용하여 객체의 위치 정보를 획득하게 된다. 따라서, 네이밍 서비스를 이용하는 클라이언트의 서비스 수행 객체에 대한 레퍼런스 획득과 서비스 요청에 따른 객체간 접속 절차는 다음과 같다.

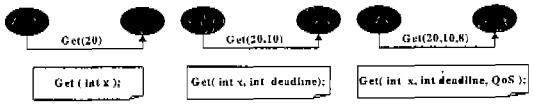
① 그룹관리자는 자신의 객체그룹에 소속되는 객체를 네이밍 서비스에 등록한다.
② 클라이언트는 원하는 서비스 객체를 네이밍 서비스에 요청한다.
③ 네이밍 서비스는 클라이언트가 요청한 서비스 객체를 등록한 그룹관리자의 레퍼런스를 클라이언트에게 반환한다.
④ 클라이언트는 네이밍 서비스에서 반환 받은 레퍼런스를 통하여 그룹관리자에게 서비스 객체의 레퍼런스를 요청한다.
⑤ 그룹관리자는 팩토리를 이용하여 서비스 객체를 생성한다.
⑥ 그룹관리자는 생성된 서비스 객체의 인터페이스를 통하여 초기화 오퍼레이션을 호출하여 서비스 객체의 레퍼런스를 반환 받는다.
⑦ 그룹관리자는 클라이언트에게 서비스 객체의 인터페이스를 반환한다.
⑧ 클라이언트는 서버에 오퍼레이션을 지금 호출할 수 있다.

위와 같은 접속 절차에 의해 클라이언트는 객체에게

서비스 수행을 요청하게 되며, 본 논문의 실시간 객체그룹에서 클라이언트와 그룹관리자와의 접속 과정상에는 네이밍 서비스를 이용하고 있음을 전제로 하고 네이밍 서비스와의 접속 과정은 생략하여 기술한다.

3.3 실시간 특성에 대한 처리 방안

비실시간 어플리케이션에 실시간 특성을 부여하려면, 여러 가지 문제점들이 발생한다. (그림 3)에서 보면, (a)는 전형적인 비실시간 어플리케이션의 오퍼레이션으로서, 서버의 오퍼레이션이 Get()으로 정수형 파라미터 하나만을 요구한다. 클라이언트는 서버의 서비스를 수행받기 위해, 서버에 정의된 오퍼레이션의 형태에 맞게 오퍼레이션을 호출하게 된다. (b)는 클라이언트가 서버에 실시간 특성인 마감시간 정보를 처리할 수 있는 오퍼레이션 호출을 하며, 서버는 오퍼레이션의 파라미터에 이를 받아 처리할 수 있도록 재정의 되어야 한다. (c)는 서버가 마감시간 뿐만 아니라 QoS에 대한 정보까지 같이 처리되기를 바란다면, 서버의 오퍼레이션은 재정의 되어야 한다. 이렇게 실시간 특성을 만족하는 서버를 만들려면, 처리될 실시간 파라미터에 따라 오퍼레이션들이 재정의 되고, 그들의 처리방안도 고려되어야 하므로, 본 논문에서는 이러한 실시간 특성에 따른 오퍼레이션들의 파라미터 정의에 대해 (그림 4)와 같은 방법을 제안하여 사용한다.



(a) (b) (c)

그림 3 비실시간 오퍼레이션에 대한 실시간 특성 파라미터들의 처리

비실시간 어플리케이션의 오퍼레이션을 실시간 특성을 지원하는 오퍼레이션으로 변환하고자 할 때에는, 각 오퍼레이션의 파라미터에 실시간 정보를 가지고 있는 구조체적인 실시간정보(본 논문에서는 RI) 파라미터만을 추가한다. RI 파라미터는 오퍼레이션이 실시간 특성을 수행하는데 필요한 파라미터 정보들이 포함되어 있으므로, 실시간 어플리케이션을 작성하는 개발자들은 지원될 실시간 특성에 대한 파라미터들을 RI에 선언해주기만 하면 된다. 실시간 객체그룹에는 실시간 정보를 처리할 실시간 관리자를 두어, 서버 객체와 클라이언트에 대한 실시간 처리 수행에 대한 투명성과 간편성을 제공한다. (그림 4)는 실시간 객체그룹에서 제시하고 있는 실시간 파라미터에 대한 선언 방법이다.

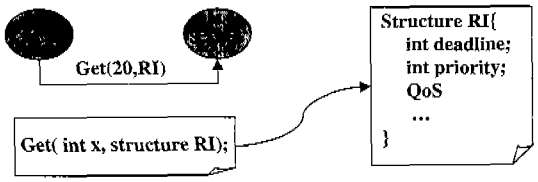


그림 4 실시간 객체그룹의 실시간 파라미터 처리

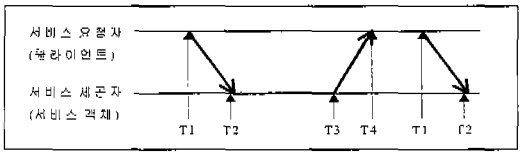
3.4 실시간 스케줄링의 시간 제약조건

분산 환경 하에서 실시간 처리를 요구하는 화상회의, 원격강의 및 각종 서비스의 시간적인 정확성을 보장하기 위해서는 응용 시스템 상에서 필요한 시간 제약조건을 명확하게 제시하여야 한다. 따라서, 이 절에서는 본 논문에서 제시한 모델에 사용되는 시간 제약조건에 대한 정의를 기술한다. 본 연구에서 사용되는 시간들은 전역 시간임을 전제로 하고, 각 시스템들이 전역 시간을 획득하는 방법에 대해서는 본 논문에서 언급하지 않는다.

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 시간 제약조건들은 다음과 같이 크게 5가지로 정의한다.

- ▸ 호출시간(IT : Invocation Time)
  - 클라이언트의 호출시간(*CIT*)
  - 서비스 객체의 호출시간(*SIT*)
- ▸ 수행시간(*ST* : Service Time)
- ▸ 전송시간(*TT* : Transfer Time)
- ▸ 서비스 수행 마감시간(*SD* : Service Deadline)
- ▸ 서비스 요청 마감시간(*RD* : Request Deadline)

IT는 호출시간으로, *CIT*는 클라이언트가 서비스 객체에게 서비스를 요구하는 시간이고 *SIT*는 서비스 객체에게 클라이언트의 요청이 도착한 시간 즉, 서비스 객체의 실제 호출시간이다. *ST*는 서비스 객체가 서비스를 수행하는데 걸리는 시간이며, *TT*는 클라이언트의 요청이 서비스 객체에 도착하는 데까지의 전송시간 즉, $SIT - CIT$를 의미한다. *SD*는 서비스 객체가 반드시 완료되어야 하는 시점이며, *RD*는 클라이언트가 서비스를 요구한 후에 수행 결과가 반환되어야 하는 시점을 의미한다. 여기에서 *RD*가 분산 실시간 처리 시스템에서 보장되어야 하는 제약조건이다. (그림 5)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 5 시간 제약조건의 정의

위에서 정의한 시간 제약조건들에 적절한 값을 예상하여 명확하게 수행되도록 하기 위해서는, 실시간 시스템의 중요한 특성 중의 하나인 예측 가능성(predictability)을 기반으로 이루어져야 한다. 먼저 예측 가능성에 필요한 시간값을 정의하기 위해서는 실행될 프로그램의 내부 구조를 분석하여야 한다. 프로그램의 코드 상에는 정확하게 값을 사전에 예측할 수 있는 바운드된 시간(Bounded Time: *BT*)과 예측할 수 없는 바운드 되지 않은 시간(Unbounded Time : *UT*)이 있다. 따라서, 시간 제약조건의 값들은 BT와 UT의 합으로 이루어진다. *BT*는 프로그램 코드 상에서 한계 있는 반복(bounded loop)을 포함한 수행될 실행문의 합으로 미리 정의될 수 있으나, *UT*는 프로그램 수행 도중에 비동기적으로 발생되어 해당되는 수행 시간을 예측할 수 없는 부분이다. 예를 들어, 서버의 상태에 따른 수행 결과의 응답시간, 자원의 충돌에 따른 지연시간, 장애 발생에 따른 예외처리 시간 등은 정확한 시간을 예측할 수 없는 요소이다. 따라서, 최악의 경우의 실행시간을 마감시간으로 사용하며, 이를 기반으로 시간 제약조건들을 위한 적절한 값을 할당한 시간 예측 규칙을 정의한다.

$TT = SIT - CIT$ (그림 5에서 $T2 - T1$)

$RD > ST$(실행시간) + (2 * 전송시간) + α(슬랙시간: *slacktime*)

= 메소드의 보장된 완료시간 + 최대 전송시간 + 호출시간

$SD = ST + \alpha < RD - TT$

서비스 실행시간인 *ST*는 예측성을 바탕으로 결정된 순수 실행시간이다. 클라이언트의 요청이 서비스 객체에 전송되는 시간은 서비스 객체에 요청이 도착된 서비스 객체 호출시간에서 클라이언트 객체의 호출시간의 차를 구하면 된다. 서비스 요청 마감시간인 *RD*는 서비스 실행시간, 통신 지연, 슬랙 시간 등의 시간을 포함한다. 서비스 수행 마감시간은 서비스 실행시간에 슬랙 시간의 합이며, 이것은 클라이언트의 서비스 요청 마감시간에서 전송시간을 뺀 값보다는 작아야 한다.

4. 실시간 객체그룹 설계

본 장에서는 실시간 객체그룹의 구조를 정의하기 위해 각 구성 요소 객체들에 대한 설계와 기능 정립 그리고 객체들의 IDL 설계를 나타낸다.

4.1 구성 요소 객체 설계

실시간 객체그룹은 분산 환경의 분산 객체들에 대한 관리의 효율성과 실시간 특성을 이용한 서비스 수행을 쉽게 하고자 한다. 따라서, 실시간 객체그룹의 구성 요소들은 관리적 기능 수행 구성 객체들과 서비스 호출 수행 구성 객체들로 나누어 기술된다. 먼저, 관리적 구성객체들로 그룹관리자(GM)는 객체그룹에 대한 관리적인 책임을 지며, 모든 객체그룹내 서비스 객체들에 대한 서비스 요청에 필요한 관리적인 절차를 객체그룹내의 다른 구성 요소 객체들 즉, 보안 객체, 객체 정보 레포지토리, 객체 팩토리와 함께 상호작용하며 수행한다. 객체 팩토리는 그룹관리자의 객체 생성 요구에 따라, 객체 정보 레포지토리의 정보를 이용하여 객체를 생성한 후, 생성된 객체의 레퍼런스를 그룹관리자에게 반환하며, 서비스 수행 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 호출 수행에 관련된 객체들은 실시간 관리자, 타이머 객체, 스케줄링 객체이다. 실시간 관리자는 분산 객체들에게 시간 제약조건의 처리에 대한 투명성을 제공하며, 서비스 객체에게 실시간 특성을 부가하여, 이에 따른 처리를 위해 스케줄러 객체, 타이머 객체와 상호 작용한다. 실시간 관리자는 서비스를 요청하는 객체(클라이언트)가 소속되어 있는 그룹 내에 있는 실시간 관리자와 서비스를 수행하는 객체가 소속되어 있는 그룹 내에 있는 실시간 관리자의 기능이 구별된다. 타이머는 클라이언트나 서비스 수행 객체에게 요청된 마감시간의 알람기능을 제공하고,스케줄러는 특정 실시간 스케줄링 알고리즘에 의해 구현된 객체이다. 스케줄러는 적용되는 스케줄링 알고리즘에 따라, 클라이언트들의 우선순위를 선정하여, 서비스 수행 객체들이 이 우선순위에 따라 서비스 요청을 받아들일 수 있도록 한다. 스케줄러는 서비스 수행 요청이 들어온 서비스 객체들마다의 스케줄링 리스트(우선순위 리스트)를 관리하여 우선순위를 스케줄링한다. 이때, 스케줄링의 대상은 지금 요청이 들

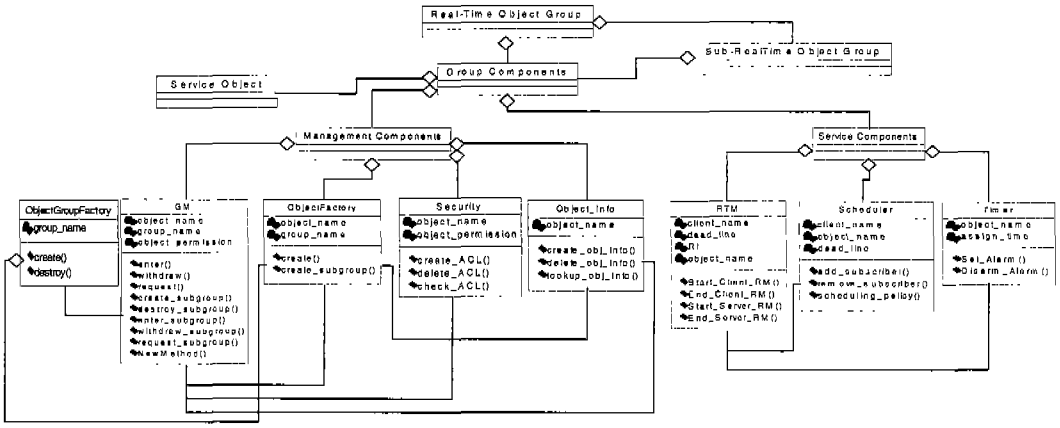


그림 6 실시간 객체그룹의 클래스 다이아그램

어온 클라이언트와 스케쥴링 리스트에 저장되어 있으면서, 서비스 수행을 받지 못한 클라이언트들이 된다.

실시간 객체그룹의 구성 요소 객체들에 대한 전체적인 클래스 다이아그램은 (그림 6)에 나타낸다.

4.2 기능 정립

실시간 객체그룹에서 처리하는 요청들에 대한 관리적인 수행 절차와 서비스 수행에 따른 절차를 객체들의 접속 절차도와 시퀀스 다이아그램으로 나타낸다.

4.2.1 관리적 절차

실시간 객체그룹의 관리적 절차는 그룹관리자를 중심으로 발생한다. 즉, 실시간 객체그룹의 모든 관리에 대한 요청은 그룹관리자를 통하여 일어나게 된다. 실시간 객체그룹의 그룹관리자를 통하여 요청할 수 있는 관리적 기능에는 분산되어 있는 서비스 수행객체를 실시간 객체그룹으로 포함, 실시간 객체그룹에 소속되어 있는 객체의 탈퇴, 클라이언트의 서비스 수행객체에 대한 레퍼런스 요청, 객체그룹내 서브객체그룹의 생성, 서브객체그룹으로의 서비스 수행객체의 포함, 탈퇴, 서브객체그룹내의 서비스 수행객체에 대한 레퍼런스 요청 등이 이에 해당된다. 이러한 관리적 기능 수행절차들 중에, 클라이언트의 서비스 객체 레퍼런스 요청에 대한 객체간의 상호 접속 절차를 예로 들어 설명하며, 이들간의 상호 접속 과정과 시퀀스 다이아그램은 각각 (그림 7)과 (그림 8)에 나타낸다.

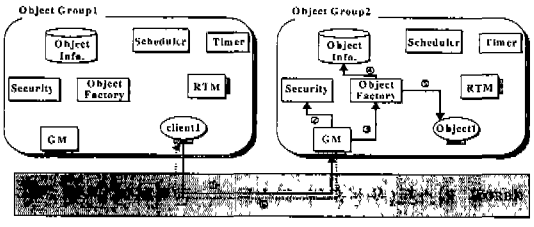


그림 7 클라이언트의 레퍼런스 요청에 대한 접속 절차도

(그림 7)은 클라이언트가 서비스 객체를 호출하기 앞서, 서비스 객체에 대한 레퍼런스를 얻는 과정이다. 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 보안객체에게 넘기고, 보안 객체는 접근 권한 검사 수행 후 결과를 반환한다. 그룹관리자는 반환 받은 결과가 클라이언트의 접근 가능이면, 객체 팩토리 객체에게 서비스 객체의 생성을 요청한다. 객체 팩토리 객체는 생성을 요청 받은

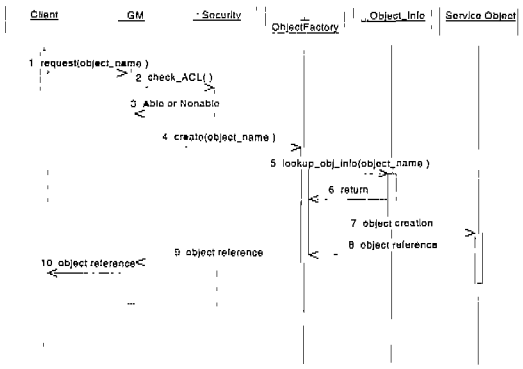


그림 8 클라이언트의 레퍼런스 요청에 대한 시퀀스 다이아그램

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.

4.2.2 서비스쪽 절차

클라이언트의 서비스 요청은 먼저 관리 절차에 따라 서비스 객체의 레퍼런스를 획득한 후에 요청된다. 이때 앞서 말한바와 같이, 실시간 특성을 가진 서비스 요청이므로 클라이언트의 오퍼레이션 호출에는 반드시 시간적 제한에 따른 표현(RI)이 포함되어져야 한다. 또한, 각 객체그룹의 실시간 관리자는 전역 시간 서비스를 이용하여 분산 환경에서의 전역 시간 정보를 계속 유지하고 있다고 전제하며, 전역 시간 서비스에 대한 세부사항은 본 연구의 대상으로 삼지 않는다.

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 실시간정보(RI)에 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 객체에게 실질적인 서비스를 요청하며, 이때 앞서 수행된 결과를 저장하고 있는 실시간정보도 같이 서비스 객체에게 넘겨진다. 서비스 객체는 넘겨 받은 실시간정보를 자신의 객체그룹에 있는 실시간 관리자에게 전달하고, 실시간 관리자는 실시간정보를 이용하여, 전송시간을 고려한 서비스 수행에 필요한 마감시간을 계산한 후, 계산된 마감시간을 스케쥴러 객체에게 넘겨주어, 클라이언트의 서비스 수행 우선순위를 스케쥴링 하게 한다. 스케쥴링의 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, 클라이언트에게 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 9)는 객체간의 접속 절차도를 나타내며,. (그림 10)은 시퀀스 다이아그램이다.

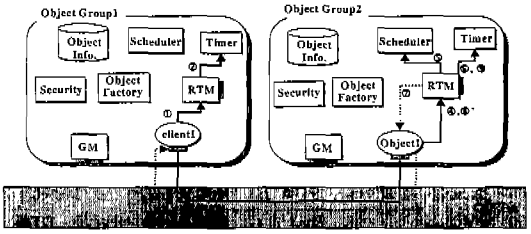


그림 9 클라이언트의 실시간 서비스 요청에 대한 접속 절차도

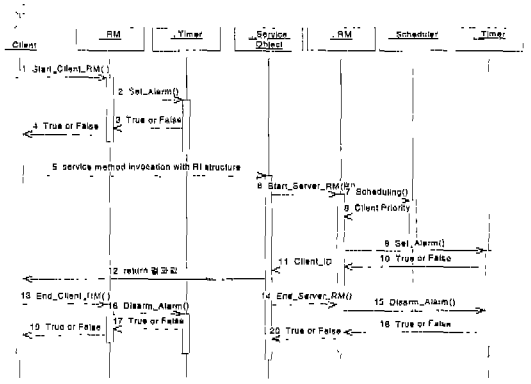


그림 10 클라이언트의 서비스 요청에 대한 시퀀스 다이아그램

(그림 11)은 클라이언트와 서버간의 실시간 서비스 요청에 따른 실시간정보 파라미터에 있는 시간 제약조건의 표현과 그것의 처리에 따른 순차적인 과정을 값에 의한 예제로 자세히 나타낸다.

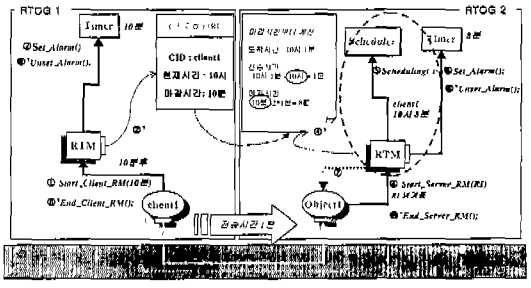


그림 11 실시간정보의 사용에 대한 절차도

① 클라이언트(Client1)는 서비스 객체(Object1)의 메소드를 호출하기 전에, 자신의 실시간 객체그룹(RTOG1)에 있는 실시간 관리자(RTM)에게 마감시간(10분) 정보를 넘겨준다.

② RTOG1의 RTM은 타이머를 마감시간(10분)으로 설정하고, 실시간정보(RI) 구조체에 실시간 처리 정보들(클라이언트 식별자, 현재시간, 마감시간의 상대값)을 저장한다.

③ ②의 절차가 끝나면, Client1은 Object1의 set()메소드를 호출한다. 이때 RI 정보도 같이 넘겨준다.

④ Object1은 Client1에게서 받은 파라미터중 RI를 자신의 실시간 객체그룹(RTOG2)의 RTM에게 넘긴다. RTOG2의 RTM은 RI의 정보를 이용하여, 요청의 전송시간(1분)과 그에 따른 상대적 마감시간(10분-왕복전송시간=8분)을 계산한다.

⑤ RTOM2의 RTM은 계산된 마감시간을 기반으로 스케줄러에게 우선순위 설정을 위한 스케줄링(예로서, EDF를 적용)을 요청한다.

⑥ RTOG2의 RTM은 우선순위에 따라 클라이언트의 계산된 상대적 마감시간값(8분)을 타이머에 설정한다.

⑦,⑧ 일련의 과정이 끝나면, Object1은 서비스를 수행한 후, 결과를 Client1에게 반환한다. 동시에, RTOG2의 RTM에게 서비스 수행의 종료를 알린다. 결과를 반환받은 Client1은 RTOG1의 RTM에게 서비스 요청의 종료를 알리고, RTM은 타이머를 해제한다. 마찬가지로, RTOG2의 RTM도 타이머를 해제한다.

4.3 IDL 설계

본 절에서는 실시간 객체그룹의 구성 요소들 중에 관리적인 기능과 서비스적인 기능의 가장 중요 역할을 차지하는 그룹관리자와 실시간 관리자 그리고 스케줄러 객체에 대한 IDL을 나타낸다.

그룹관리자는 실시간 객체그룹의 관리적인 측면으로 오퍼레이션들을 수행한다. 즉, 실시간 객체그룹 생성자가 특정한 기준에 맞는 실시간 객체그룹내에 객체들을 소속시킬 수 있도록 하는 enter()와 소속되어 있는 객체들을 탈퇴시킬 수 있는 withdraw() 오퍼레이션을 제공한다. 또한 실시간 객체그룹내에 소속 그룹으로 서브그룹을 생성할 수 있도록 하는 create_subgroup()오퍼레이션과 그 서브그룹에 객체를 소속시키거나 탈퇴시킬 수 있는 각각의 enter_subgroup()과 with-draw_subgroup()오퍼레이션도 가지고 있다. 또한 객체들의 서비스를 요청하기 위해 클라이언트가 먼저 수행해야한 일련의 과정중 그룹관리자를 통한 서비스 객체/서브그룹내 서비스 객체의 레퍼런스 획득에 필요한 request()오퍼레이션과 request_subgroup() 오퍼레이션을 제공한다. (그림 12)은 실시간 객체그룹의 그룹관리자에 대한 IDL의 일부를 나타낸다.

```
interface GroupManager{

    typedef  string                 Object_Name;
    typedef  string                 Object_Permission;
    typedef  string                 Group_Name;

   /* 객체 관리 서비스 */
    boolean enter(in Object_Name obj_name, in
    Object_Permission obj_permission);
    boolean withdraw(in Object_Name obj_name, in
    Object_Permission obj_permission);
    string request(in Object_Name obj_name);
    ......
}
```

그림 12 그룹관리자 객체의 IDL

실시간 관리자는 실시간 객체그룹에서 실시간적인 특성 즉, 시간 제약조건에 대한 전반적인 처리를 수행한다. 따라서, 클라이언트측 실시간 객체그룹에서 수행되는 Start_Client_RM()과 End_Client_RM(), 서버측 실시간 객체그룹에서 수행되는 Start_Server_RM()과 End_Client_RM()오퍼레이션을 가진다. Start_Client_RM ()은 클라이언트의 마감시간 정보를 실시간정보 파라미터에 저장하고, 클라이언트측 타이머의 알람을 설정하는 일련의 과정을 수행한다. End_Client_RM()은 서버 측으로부터 서비스 요청에 대한 결과가 반환되었을 때, 설정했던 알람을 해제하는 기능을 수행한다. Start_Server_RM()은 서비스 객체가 클라이언트로부터 서비스 요청을 받았을 때, 같이 전달된 실시간정보를 처리하도록 실시간 관리자에게 요청하는 오퍼레이션으로서, 실시간정보의 시간값을 추출하여 전역시간에 맞는 마감시간을 계산하여, 스케쥴러에게 스케줄링을 요구하고, 반환 받은 결과에 따라 타이머의 알람을 설정하며, 서비스 객체에 우선순위가 높은 클라이언트의 정보를 반환하는 일련의 과정을 수행한다. End_Server_RM()은 서비스 객체가 서비스 수행을 완료하고 클라이언트에게 결과를 반환할 때, 설정된 타이머를 해제하고 새로운 클라이언트의 요청을 받아들이는 일들을 수행한다. (그림 13)은 실시간 관리자에 대한 IDL을 나타낸다.

스케쥴러는 실시간 관리자의 요청에 따라 적용되는 스케쥴링 정책에 맞게 클라이언트들의 우선순위를 결정한다. 어플리케이션의 특성에 맞는 알고리즘을 상황에 따라 구현할 수 있는 스케쥴링 정책 적용 오퍼레이션으로 scheduling_policy()를 두어 개발자가 환경에 적합한 알

```
interface RTM{
    typedef  string                    Client_Name;
    typedef  string                    Object_Name;
    typedef  long                                  Time;
    struct RI{
            Client_Name  clt_name;
            Time  current_time;
            Time  c_dead_line;
    }
    Start_Client_RM(in Client_Name  clt_name, in Time
    dead_line);
    End_Client_RM(in Client_Name  clt_name);
    Start_Server_RM(in Object_Name  obj_name, in RI);
    End_Client_RM(in Object_Name  obj_name);
}
```

그림 13 실시간 관리자 객체의 IDL

고리즘을 적용시키기 쉽도록 했다. 실시간 관리자는 새로운 클라이언트의 서비스 요청이 들어오면, add_subscriber()오퍼레이션을 이용하여 클라이언트들의 마감시간을 이용하여 우선순위를 결정하도록 스케줄러에게 요구한다. 스케줄러는 적용된 스케줄링 알고리즘에 따라 클라이언트들의 우선순위를 결정하여, 우선순위 리스트에 저장하며 가장 우선순위가 높은 클라이언트의 정보를 실시간 관리자에게 넘긴다. 실시간 관리자는 새로운 클라이언트의 요청은 없지만, 서비스 객체로부터 서비스 수행이 완료되었다는 신호가 들어오면, 스케줄러에 큐되어있는 클라이언트들 중에 가장 우선순위가 높은 클라이언트에 대한 정보를 스케줄러에게 요구하며, 이때 remove_subscriber()오퍼레이션을 이용한다. 이 오퍼레이션이 호출되었을 때에는 스케줄링 알고리즘에 의한 스케줄링은 발생하지 않고, 단순히 우선순위 리스트로부터 가장 위에 있는 정보가 실시간 관리자에게 반환되며, 반환된 클라이언트의 정보는 리스트에서 삭제된다. 그러나, add_subscriber()오퍼레이션이 호출되었을 때에는 우선순위 리스트에 있는 클라이언트들과 새롭게 들어온 클라이언트와의 전체적인 스케줄링이 일어나며, 그 결과로서 우선순위 리스트의 항목들의 순서가 변경되게 된다. 스케줄러 객체의 IDL은 (그림 14)에 나타낸다.

```
interface Server{
        void receive_update();
};
typedef sequence<Server> Client_List;
interface Scheduler{
    typedef  string                    Object_Name;
    typedef  long                      Time;
    boolean add_subscriber(in Server s, in Object_Name
    obj_name, in Time dead_line);
    void remove_subscriber(in Server s, in Object_Name
    obj_name);
    void scheduling_policy(in Time dead_line);
};
```

그림 14 스케줄러 객체의 IDL

또한, 본 실시간 객체그룹에서 스케줄러의 마감시간 위반에 대한 스케줄링 기준을 두 가지 설정하여 구현한다. 스케줄러가 스케줄링 할 때에, 이미 마감시간을 위반한 클라이언트들과 마감시간을 위반하지는 않았지만 남아있는 시간이 서비스 객체의 예측되는 수행시간보다 작을 때에는 우선순위 리스트에 저장되지 않고, 실시간 관리자에게 마감시간이 위반됨을 알려주게 한다.

5. 구현과 검증

본 장은 실시간 객체그룹 플랫폼의 구현 환경과 구성 요소들의 수행 검증 과정을 나타낸다.

5.1 구현 환경

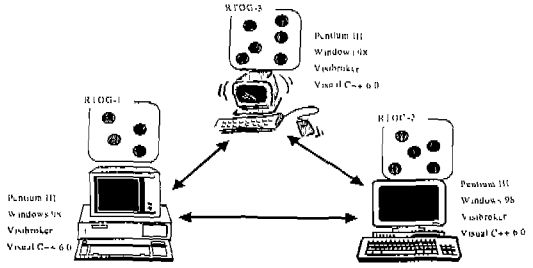


그림 15 개발 환경

본 실시간 객체그룹이 개발된 모델의 환경은 (그림 15)와 같다. 각 시스템에서 실시간 객체그룹 플랫폼을 이용하여 객체들간의 관리와 서비스에 대한 일을 수행한다.

5.2 수행 검증

본 절에서는 실시간 객체그룹의 서비스적인 측면에서, 3 클라이언트가 서비스 요청을 하였을 때 스케줄러 객체와 실시간 관리자 객체의 수행을 검증하는 과정이다. 3 클라이언트가 각각 5번의 서비스 요청을 하기 때문에, 한 서비스 객체에게 들어오는 클라이언트의 요청은 모두 15번에 해당된다. (그림 16)은 3 클라이언트의 요청에 대한 스케줄러 객체의 수행화면이다. 화면 캡쳐에서 보이는 것처럼 클라이언트의 우선순위가 스케줄러의 스케줄링 정책의 기준에 따라 변화됨을 볼 수 있다. 본 시뮬레이션에서는 EDF 스케줄링 알고리즘을 적용시켰기 때문에, 마감시간이 짧은 클라이언트의 서비스 요청이 우선순위가 높게 된다. 스크립트 화면에서 보면 클라이언트의 요청이 스케줄링 리스트에서 삭제가 되는 경우는 마감시간이 위반된 경우와 아직 마감시간이 위반되지 않았지만 남아있는 시간이 현재 서비스 수행에 걸리는 시간보다 적게 남아있는 경우들이다.

그림 16 3-콜라이언트에 대한 스케쥴러의 수행

(그림 17)은 실시간 관리자의 수행 화면을 나타내며, 클라이언트_1의 5번 서비스 요청에 대해 2번의 서비스 성공과 3번의 서비스 실패, 클라이언트_2의 5번 서비스 요청에 대해 4번의 서비스 성공과 1번의 서비스 실패, 클라이언트_3의 5번 서비스 요청에 대해 5번 모두의 성공을 나타낸다.

그림 17 3-클라이언트에 대한 실시간 관리자의 수행(1)

6. 결 론

최근 사용자에게 분산 환경에 대한 투명성을 제공하고 어플리케이션의 생산성을 향상시키는 분산 객체 컴퓨팅은 이종의 분산 환경에서 여러 종류의 어플리케이션을 통합할 방식의 필요성을 야기했다. 이러한 요구를 만족시키기 위해 RPC나 OLE가 등장하였으나, 여러 가지 문제점을 야기시켜 새로운 공통구조의 개념의 필요성이 대두되었다. 이에 맞추어 분산 객체 시스템의 설계와 개발에 대한 표준으로 제시된 것이 CORBA와 TINA이다.

분산 환경에서 분산 어플리케이션을 개발하는 경우는 어플리케이션에 의해 사용되는 데이터가 분산되어 있거나, 어플리케이션의 연산이 분산되어 있거나 또는 어플리케이션을 사용하는 사용자가 분산되어 있을 때이다. 이러한 분산 어플리케이션 개발자에게 어플리케이션을 개발할 때 분산된 객체들의 관리와 접근에 대한 용이성을 제공하기 위해 객체그룹의 개념이 대두되었다. 객체그룹은 분산 객체 컴퓨팅 환경인 TINA에서 개념을 기술해 놓고 있어, 이 개념을 도입하여 CORBA기반의 객체그룹 구조를 이전 연구[14,15]에서 제시했다.

또한 분산 어플리케이션은 실시간적 특성에 대한 다양한 요구사항을 지원할 필요가 생기고 있다. 실시간 특성 지원의 가장 첫째 요소인 시간제약조건에 대한 표현과 이에 대한 처리가 수행되어져야 한다.

그러므로, 본 논문은 분산 환경인 표준 CORBA에 객체들의 관리를 용이하게 하고, 실시간 특성인 시간 제약조건등을 처리할 수 있는 실시간 객체그룹 플랫폼 구조를 제시한다. 기존에 이루어지고 있는 실시간 분산 환경은 실시간 CORBA를 사용하거나 표준 CORBA 상에 ORB를 수정하거나 확장하는 방법으로 새로운 실시간 CORBA 환경을 만들고 있다. 이러한 환경은 특정한 운영체제나 실시간 CORBA 제품군을 사용하여, 전체적인 시스템의 성능보다는 특정한 부분에 대한 성능 향상에 초점을 맞추고 있다. 그러므로, 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 기능에 따른 검증을 보여주었다.

실시간 객체그룹 플랫폼은 표준 CORBA환경에서 ORB의 수정 없이 시간 제약조건의 표현과 처리가 간단해지며, 클라이언트나 서비스 객체에게 실시간 처리에 대한 투명성이 제공되며, 실시간 특성에 대한 정보 파라미터들을 확장 가능하여 유연성을 제공하고, 이러한 확장성이 클라이언트나 서비스 객체와 전혀 무관하게 이루어질 수 있다. 실시간 객체그룹은 객체의 관리에 대한 그룹관리자와 실시간 처리 즉, 서비스에 대한 실시간 관리자를 각각 두어 분산 객체들의 효율적인 관리와 실시간 특성에 대한 클라이언트와 서버의 투명성을 제공한다. 이를 위해, 실시간정보라는 구조체를 정의하여 실시간 파라미터 등을 처리하였고, 타이머객체를 두어 클라이언트와 서버 각각의 마감시간 확인을 용이토록 했다.

또한 스케쥴러 객체를 따로 두어, 본 연구의 플랫폼을 사용하는 시스템의 상황에 맞는 또는 어플리케이션의 특성에 맞는 스케쥴링 알고리즘을 선택하여 유동적으로 사용할 수 있도록 하였다.

추후 연구로는 실시간정보에 포함되는 파라미터의 종류들을 확장하여, 단순한 마감시간이 아닌 클라이언트의 중요도에 따른 우선순위나 병행 제어에 따른 정보, 멀티미디어 처리에 대한 QoS 파라미터 등을 확장시켜 적용해본다. 또한, 스케쥴러가 사용하는 스케쥴링 정책을 여러개두어 클라이언트가 전달한 파라미터의 특성에 따라 스케쥴링 정책을 선택할 수 있도록 하는 구조로 보완하고자 한다.

참 고 문 헌

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

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

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

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

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

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

김 명 희

1993년 원광대학교 컴퓨터공학과 공학사. 1996년 원광대학교 컴퓨터공학과 공학석사. 2001년 원광대학교 컴퓨터공학과 박사. 2001년 ~ 현재 원광대학교 전기전자·정보공학부 BK21 계약교수. 관심분야는 분산 실시간 컴퓨팅, 시스템 최적화, 운영체제

주 수 종

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 컴퓨터·정보통신공학부 교수. 관심분야는 멀티미디어 데이터베이스, 분산 실시간 컴퓨팅, 시스템 최적화, 분산객체모델