능동 복제 기반 CORBA 객체 그룹 지원

손 덕 주[†] · 신 범 주[†] · 남 궁 한[†] · 진 성 일[††]

요 약

분산 객체 시스템에서 객체 그룹의 지원은 부하 분산, 고장 감내 그리고 높은 가용성과 같은 장점을 제공한다. 본 논문에서는 능동 복제에 기반하여 객체 그룹을 지원하는 객체 중개자의 개발에 대해 기술한다. 본 논문에서 기술되는 객체 중개자는 클라이언트와 객체 그룹간의 통신에는 IIOP를 사용하며, 객체 그룹들 사이의 일치성 제어를 위하여 전체 순서를 지원하는 다중 전송 프로토콜을 사용한다. 또 객체 그룹을 지원하기 위하여 필요한 기능들을 객체 중개자를 확장하는 방법으로 제공한다. 본 논문의 객체 중개자는 서버 객체 중복의 투명성을 제공함으로써 기존 CORBA와 상호 운용될 수 있다. 또 추가되는 응용 프로그래밍 인터페이스 함수를 최소화함으로써 기존의 서버 응용을 객체 그룹을 지원하는 서버 응용으로 쉽게 확장할 수 있게 한다. 프로토타입을 구현하고 성능 시험을 통하여 단일 객체 호출과 객체 그룹에 대한 호출의 성능을 비교하였다.

Supporting CORBA Object Group based on Active Replication

Duk-Joo Son[†] · Bum-Joo Shin[†] · Han Namgoong[†] · Seung-Il Jin[††]

ABSTRACT

Supporting object group on distributed object system gives merits such as load balancing, fault tolerance and high availability. In this paper, we describe a CORBA ORB that has been designed to support object group based on active replication. The ORB supports the operational model in which it uses the IIOP for communication between client and server and total ordered multicast protocol for consistency control among group members. And through extension of ORB, it provides functions required for support of object group. Since it provides transparency of object replication, the ORB is interoperable with the existing CORBA products. It make possible for existing server application to be easily extended to application supporting object group as adding interface functions which should be used for building applications is minimized. A prototype is implemented, and performance of the replicated object group is tested and compared with a single object invocation.

1. 서 론

분산 객체 시스템에서의 객체 그룹은 중복된 여러 객체들이 외부적으로 마치 하나의 객체처럼 동작하는 것을 의미한다. 객체 그룹은 그룹 멤버들 사이에 일치된 상태를 유지시키는 방법에 따라 능동 복제(Active Replication) 와 수동 복제(Passive Replication)로 구분한다. 능동 복제는 상태 기계(State Machine)[16]에 따라 일치성이 유지된다. 즉 객체 그룹을 구성하는 모든 멤버가 동일한 상태에서 시작하여, 동일한 순서로 오퍼레이션을 수행할 경우 항상 동일한 상태에 도달 할 수 있다는 것이다. 능동 복제는 수행 시에 많은 자원이 요구된다는 단점에도 불구하고 부하 분산 및 높은 가용성을 제공하며, 오류 복구 시간이 짧다는 특성 때문에 많은 응용에 이용될 수 있다.

CORBA는 분산 객체를 지원하기 위한 산업체 표준이며, 하나의 클라이언트는 하나의 서버에 연결되는

† 정 회 원 : 한국전자통신연구원 인터넷서비스연구부 책임연구원
†† 종신회원 : 충남대학교 컴퓨터과학과 교수
논문접수 : 1999년 10월 14일, 심사완료 : 1999년 11월 12일

원격 호출 함수(remote procedure call)를 기반으로 한다[14]. 따라서 CORBA에 객체 그룹을 지원하기 위해서는 클라이언트의 함수 호출이 그룹의 멤버가 존재하는 모든 노드에서 수행될 수 있는 구조로 확장하여야 한다. 이러한 구조로 확장하는 방식은 가로채기 방식(Interception Approach)[9], 서비스 방식(Service Approach)[11] 그리고 통합적 방법(Integration Approach)이 있다. 가로채기 방식은 복제 기능을 클라이언트에 투명하게 제공할 수 있다는 장점을 가지고 있으나 가로채기 기능이 시스템에 종속적이라는 단점이 있다. 서비스 방식은 CORBA의 기본 개념에 적합하다는 장점을 가지는 반면 기존의 클라이언트를 지원할 수 없다는 단점을 가진다. 반면 통합적인 방법은 객체 중개자(ORB : Object Request Broker)[14]의 내부 구조를 변경하여야 한다는 점이 있으나 클라이언트 투명성을 제공할 수 있으며, 적절한 동작 구조를 제공함으로써 기존 응용들과 상호 호환성을 제공할 수 있다는 장점을 제공한다.

객체 그룹의 멤버들이 하나의 객체처럼 동작하기 위해서는 멤버들이 동일한 상태를 유지할 수 있도록 하여야 한다. 특히 능동 복제 방식에서는 각 멤버들에 도달하는 메시지의 순서를 일치시켜 상태를 일치시키는 방법을 사용하기 때문에 다중 전송 채널을 사용한다. 이 때 다중 전송 채널을 CORBA 구조에 연결하는 방법도 설계 시에 고려되어야 할 중요한 사항이다. 두 가지 구조를 고려할 수 있는데, 한 가지 구조는 클라이언트 및 객체 그룹이 다중 전송 채널을 공유하는 것이며, 다른 한 가지는 클라이언트와 그룹의 특정 멤버 사이에는 기존 단일 전송 채널을 사용하고 그룹 멤버들 사이의 일치성을 제어하기 위하여 다중 전송 채널을 사용하는 구조이다.

본 논문에서는 객체 중개자를 확장하는 통합적인 방식을 이용하여 객체 그룹을 지원하는 방법을 사용하며, 클라이언트는 단일 전송 채널로 객체 멤버에 접속하는 구조로 동작하는 객체 그룹 지원 CORBA에 대해 기술한다.

본 논문의 다음 장에서는 기존 연구를 분석하며, 3장에서는 객체 그룹 지원을 위한 동작 구조를 기술한다. 제 4장에서는 클라이언트가 객체 그룹에 접속하기 위하여 사용하는 객체 참조 구조에 대해 기술하고, 5장에서는 그룹 관리를 기술한다. 6장에서는 원격 함수 호출의 확장을 다루며, 7장에서는 오류 발생시의 복구 과정을 기술한다. 8장에서는 상호 운용성에 대해 언급하며, 9장에서는 성능에 관해 분석하고 10장에서 결론을 맺는다.

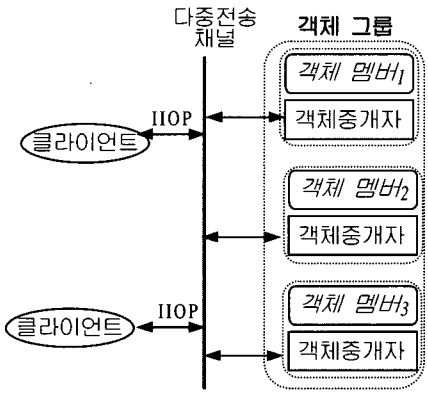
2. 기존 연구

앞 장에 기술한 바와 같이 기존 연구를 객체 그룹 지원 방식에 따라 각각의 장단점을 갖는다. 기존 연구를 이러한 관점에서 나누어 보면 다음과 같다. 서비스 방식을 사용한 시스템으로는 OGS[7], Newcastle 대학에서 개발한 시스템[11]이 있으며, 가로채기 방식을 이용하여 구현된 결과물[6, 12]들도 있다. 본 논문에서는 객체 중개자의 확장을 통해 객체 중복을 지원하며, 클라이언트와 서버 사이에는 IIOP(Internet Inter-ORB Protocol)를 사용하고 중복된 서버들 사이에 일치성을 유지하기 위한 방법으로 전체 순서를 지원하는 다중 전송 프로토콜을 사용한다는 점에서 기존 연구와 차별된다. 따라서 본 논문의 방식에서는 프로그래밍 인터페이스를 단순화시켜 쉽게 객체 그룹 지원 응용으로 변환할 수 있게 할 뿐 아니라 기존 응용과의 호환성 뿐 아니라 중복 투명성도 쉽게 지원한다는 장점을 제공하고 있다.

OMG에서는 표준 제정을 위한 작업을 진행하고 있다. 현재 5개의 제안서[5, 6, 8, 13, 15]가 제출되었으며, 제안서를 제출한 기관들 간에 단일 제안서를 만들어 제출하였다. 제안서들 중에는 구현과 관련된 내용을 기술한 것도 있지만 표준의 특성상 소프트웨어의 구조, 프로그래밍 인터페이스 및 기능에 국한되어 기술되어 있다.

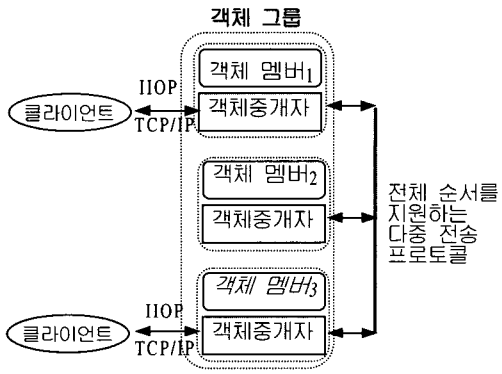
3. 동작 구조

기존의 단일 전송 채널(TCP/IP)을 사용하던 객체 중개자를 다중 전송 프로토콜(multicast protocol)을 사용할 수 있도록 확장하는 방식은 두 가지가 가능하다. 한 가지는 클라이언트가 함수 호출 메시지를 다중 전송 채널을 통해 객체 그룹 멤버들에게 보내는 구조이며, 다른 한 가지는 기존의 방식과 동일하게 클라이언트가 서버 객체 그룹의 멤버들 중 하나에 접속하여 함수 호출 메시지를 보내고, 클라이언트의 메시지를 받은 객체 그룹의 멤버가 그룹 멤버들에게 전달하는 구조이다. (그림 1)과 (그림 2)는 가능한 두 가지 구조를 나타낸다.



(그림 1) 다중 전송 채널을 공유하는 구조

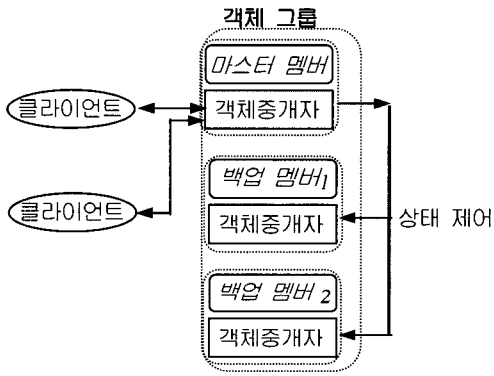
(그림 2)의 구조는 (그림 1)의 구조에 비해 클라이언트가 접속한 서버 객체에 결함이 발생하였을 때 특별한 복구 과정이 필요한 단점이 있는 반면 많은 장점을 제공하고 있다. 두 번째 구조는 서버와의 통신을 위하여 IIOP를 사용하기 때문에 기존 CORBA와의 상호 운용성이 쉽게 지원된다. 또 기존의 제품을 쉽게 확장할 수 있을 뿐 아니라 멀티캐스트 통신 프로토콜이 클라이언트에 감추어지기 때문에 멀티캐스트 프로토콜에 독립적으로 구현될 수 있다는 장점을 갖는다. 프라이머리-백업(primary-backup) 방식의 중복도 쉽게 구현할 수 있다는 점, 기존의 CORBA 응용에서 발전할 수 있는 응용들을 쉽게 지원할 수 있다 점도 이 구조가 갖는 장점이다. 예로써 thin client 또는 mobile host를 사용하는 환경을 고려할 때 client는 단순한 객체 중개자를



(그림 2) 객체 그룹을 지원하는 동작 구조

지원하는 것이 바람직하다[21].

이 같은 장점들을 지원하기 위하여 본 논문에서는 (그림 2)의 동작 구조를 사용한다. 클라이언트는 객체 참조 구조(object reference)에 나타난 서버 객체들 중 하나를 선택하여 접속한 후, 함수 호출 요구 메시지를 보낸다. 요구 메시지를 받은 서버 객체는 이를 다중 전송 채널을 통해 중복된 그룹 멤버 객체들에 전달하여 각 객체에서 동일한 오퍼레이션이 수행되게 함으로써 일치된 상태를 유지한다. 이 같은 구조는 프라이머리-백업(primary-backup) 방식의 복제도 쉽게 구현할 수 있게 한다.



(그림 3) 수동 복제 방식의 동작 구조

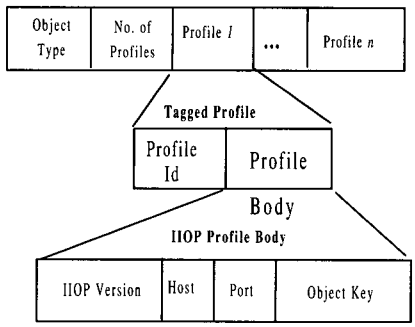
(그림 3)의 구조는 (그림 2)의 구조와 매우 유사함을 보이고 있다. 차이점은 전자의 경우 클라이언트의 함수 호출 요구 메시지가 다중 전송 프로토콜에 전송된 후 모든 노드에서 동일하게 수행되어야 하는 반면 후자의 경우 주기적으로 마스터가 백업들의 상태를 일치시키기 위하여 다중 전송 프로토콜을 사용한다는 점이다.

4. 객체 그룹 접속

CORBA에서는 클라이언트가 서버에 접속하기 위해서 사용되는 IOR(Interoperable Object Reference)이라는 객체 참조 구조를 정의하며, IIOP는 IOR에 포함되는 정보에 대하여 구체적으로 정의한다. 즉 IIOP에서는 IOR이 노드, 포트 그리고 서버 객체를 구분 할 수 있는 키(key)에 대한 정보를 제공하도록 정의하고

있다[14]. 앞서 기술한 바와 같이 본 논문의 클라이언트도 IIOP를 이용하여 서버와 통신하기 때문에 IIOP에서 정의한 객체 참조 구조를 사용하여 서버 객체 그룹의 멤버들 중 하나에 접속하여야 한다. 이를 위하여 본 논문에서는 IIOP에서 사용하는 객체 참조 구조를 변경하지 않고, 단지 의미를 확대하여 사용한다.

객체 그룹을 표현하는 객체 참조 구조는 여러 개의 프로화일(profile)을 가지며, 각각의 프로파일은 각 그룹 멤버에 관한 정보를 표현한다. 또 객체 타입(object type)을 표현하는 필드에 특정 표시어를 추가하여 객체 중개자가 객체 그룹을 나타내는 IOR임을 구분할 수 있게 한다. (그림 4)는 본 논문의 객체 그룹 참조 구조를 나타내고 있다. IIOP에서는 객체 참조 구조에 적어도 하나 이상의 프로화일을 가져야 한다고 정의되어 있기 때문에 이 구조는 CORBA IIOP에서 정의된 객체 참조 구조와 동일한 것으로 인정될 수 있다.



(그림 4) 객체 그룹 참조 구조

일반적으로 클라이언트가 (그림 4)의 객체 참조 구조를 사용하여 서버 객체 그룹에 접속할 때 첫 번째 프로화일의 객체에 접속한다. 모든 클라이언트가 이러한 방법으로 서버에 접속할 경우 첫 번째 멤버에 과부하가 발생한다. 이를 해결하기 위하여 클라이언트가 서버 객체에 접속을 요구할 때 클라이언트 객체 중개자는 객체 참조 구조에 표현된 프로화일들 중 무작위로 선택된 하나의 프로화일 정보를 이용하여 세션을 생성하고 유지한다. 그러나 이러한 접속 방법은 완전한 부하 분산을 실현하기에는 불충분하기 때문에 서버의 부하 정도에 따라 접속할 서버를 선택하는 부하 분산 메커니즘이 추가되는 것이 바람직하다.

5. 그룹 관리

객체 그룹 멤버들의 상태를 효율적으로 일치시키기 위해서는 다중 전송 프로토콜을 사용하는 것이 필요하다[10]. 객체 그룹을 다중 전송 프로토콜과 연결하기 위한 방법은 크게 두 가지가 있다. 첫째는 다중 전송 프로토콜에서 지원하는 그룹과 각 객체 그룹을 일 대 일로 연결하는 것이며, 다른 방법은 모든 객체 그룹들이 하나의 IP 다중 전송 그룹으로 모든 메시지를 공유하는 것이다. 전자의 경우는 객체 중개자에서 객체 그룹의 멤버를 관리할 필요가 없다는 장점을 가진다. 그러나 IP 기반 다중 전송 프로토콜에서 그룹은 특정 IP 주소(IP multicast address)와 연결되기 때문에 특별히 전송 계층에서 서브 그룹을 지원하지 않으면 하나의 객체 그룹이 구성될 때 마다 새로운 포트가 할당되어야 하는 단점이 있다. 수 많은 객체 그룹들이 생성될 수 있는 경우 이 같은 단점은 심각한 성능 저하를 야기할 수 있다. 후자의 경우는 전자의 단점을 해결할 수 있는 반면 객체 중개자 레벨에서 멤버들을 관리하여야 할 뿐 아니라 모든 메시지들이 객체 중개자에 전달되기 때문에 필요한 메시지만을 선별하는 작업이 필요하다. 또한 서로 관련성이 없는 메시지들도 전체 순서화의 대상이 되기 때문에 성능을 저하시킬 수 있다. 향후의 다중 전송 프로토콜은 효율적인 서브 그룹을 제공할 것으로 전망되나 현재의 환경에서는 객체 그룹이 생성될 때 마다 포트를 할당하는 것이 불가능한 것으로 판단되어 본 논문에서는 후자의 방법을 사용한다.

5.1 동적 멤버 관리

서버 프로그램에서는 객체를 생성한 후 CORBA 객체로 만들기 위해서는 해당 객체를 인터페이스 함수를 통해 객체 중개자에 등록하여야 한다. 이와 동일하게 객체 그룹의 멤버는 인터페이스 함수를 통해 객체 중개자에서 관리하는 객체 그룹의 멤버로 등록하게 된다. 본 논문에서는 객체 그룹을 생성하기 위하여 명시적인 인터페이스를 제공하지 않으며, 객체 그룹의 멤버로 등록할 때 해당 그룹이 존재하지 않을 경우 생성하는 방법을 사용한다. 이를 위하여 기존의 obj_is_ready 인터페이스 함수[14]와 유사한 obj_group_is_ready 인터페이스 함수를 제공한다. (그림 5)는 obj_group_is_ready 인터페이스를 사용한 객체 그룹을 위한 응용 프로그램을 나타낸다.

```
package benchmark;
public class Server{
    public static void main( String[] args ){
      ORB orb = ORB.init();
      BOA boa = orb.BOA_init();
      Object b = boa.create( new benchImpl(),
                    "IDL:benchmark/bench:1.0");
        . . .
      boa.obj_group_is_ready(b, "BenchmarkTest");
        . . .
      boa.impl_is_ready();
    }
}
```

(그림 5) 응용 프로그램 예

상기 프로그램은 JacORB[3]의 성능 시험 프로그램을 객체 그룹 호출의 성능 시험을 위해 수정한 것이다. 멤버 객체는 이 함수를 통해 동적으로 그룹에 가입할 수 있다. 객체 그룹은 이 함수를 통해 제공되는 그룹 명으로 구분되며, 모든 노드의 객체 중개자는 자신과 관련된 그룹 및 멤버 정보를 관리한다. 새로운 멤버가 가입될 때 기존 멤버에 대한 정보 및 상태 정보는 마스트에 의해 전송된다. 마스트는 멤버 가입 순서에 의해 결정되며, 마스트에 문제가 발생할 경우 가입 우선 순위에 의해 다음 마스트가 결정된다. 마스트는 기존 멤버들로부터 준비 완료 메시지를 받은 후 멤버 가입을 요청한 노드에 멤버 정보 및 상태를 전송한다.

5.2 상태 전달

능동 복제 방식에서는 동적 멤버 관리를 지원하기 위해서는 새로운 멤버가 가입될 때 기존 멤버와 상태를 일치시켜야 한다. 상태를 일치시키기 위해서는 클라이언트가 기존 멤버들에게 보내는 함수 호출 메시지를 제어하여야 하며, 객체의 상태를 메시지로 전달하고, 메시지로부터 객체의 상태를 읽고 저장하는 방법이 제공되어야 한다.

본 논문에서는 함수 호출 메시지를 제어하기 위하여 상태 전달 과정 동안 메시지를 저장하는 버퍼를 사용하며, 객체의 상태를 메시지에 저장하고 복구하기 위한 방법으로 Java에서 제공하는 Serializable 인터페이스를 이용한다. 본 논문의 CORBA 객체는 Java Serializable 인터페이스를 상속하기 때문에 객체 중개자에서는 Java에서 제공되는 프로토콜에 따라 객체의 상태를 메시지에 저장 및 복구한다. 따라서 효율성을 높이기 위하여 서버 객체 구현 시에 전달될 필요가 없는 멤버 변수는 transient 변수로 선언하는 것이 바람직하다. 단 transient 변수를 가진 객체는 멤버 가입이 완료된 후에 transient 변수를 새로이 초기화하여야 한다.

6. 원격 함수 호출

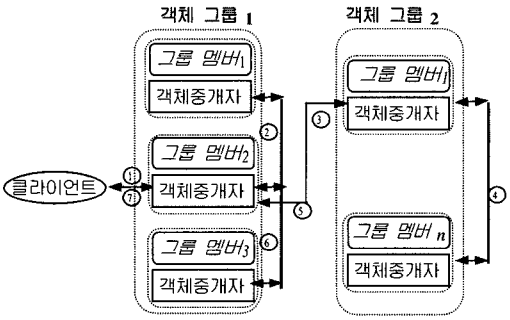
CORBA에서 지원하는 원격 함수 호출은 클라이언트와 서버 사이에 메시지의 교환이 필수적이다. 즉 클라이언트가 요청한 원격 함수 호출 메시지에 대해 서버 객체의 응답 메시지가 반드시 존재해야 한다. 이러한 시맨틱이 객체 그룹에 그대로 적용될 경우 클라이언트는 각 멤버로부터 중복된 여러 개의 응답 메시지를 받는다. 이 같은 문제를 다중 응답이라 한다. 또 다단계 클라이언트-서버 구조를 가지는 응용의 경우에는 여러 번의 중복된 함수 호출이 일어날 수 있다. 즉 서버 객체의 멤버 함수가 수행 중에 다른 서버 객체의 멤버 함수를 호출하는 경우 모든 멤버들이 요구 메시지를 보내게 되면 서버 객체에서는 멤버 수만큼의 호출 요구를 받게 된다. 이를 다중 호출이라 한다. 이 같은 경우들은 원한지 않는 결과에 도달할 수 있게 하기 때문에 방지되어야 한다.

본 논문에서는 이를 해결하기 위하여 매우 단순한 방법을 사용한다. 클라이언트가 접속된 멤버만이 클라이언트에 응답 메시지를 보낼 수 있을 뿐 아니라 외부 서버 객체에 호출 요구 메시지를 보낼 수 있게 한다. 본 장에서는 다중 응답 및 다중 호출을 방지하는 방법에 대해 기술한다.

6.1 다중 응답 방지

다중 응답을 방지하기 위하여 본 논문에서는 프라이머리-백업과 유사한 방법을 사용한다. 자신의 노드에 접속된 클라이언트에 의해 발생한 요청 메시지에 대해 클라이언트가 접속된 멤버 객체 만이 응답할 수 있으며, 나머지 멤버 객체들은 응답 메시지를 보내지 않는다. (그림 6)에서 객체 그룹 1의 경우에는 클라이언트의 함수 호출 요구에 대해 멤버 2만이 응답하게 된다. 실제 객체 그룹의 다른 멤버들은 클라이언트와의 세션을 갖고 있지 않기 때문에 결과 값을 보낼 수 있는 방법이 없다. 이 같은 동작 방식을 지원하기 위하여 클라이언트로부터 보내진 IIOP호출 메시지는 메시지의

변경 과정을 거쳐 다중 전송 채널에 보내어 진다. 다중 전송 채널을 통해 전달되는 호출 요구 메시지는 IIOP의 request 메시지의 정보 및 request 메시지를 야기시킨 클라이언트와 관련된 정보를 포함한다.



(그림 6) 객체 그룹의 외부 객체 호출 과정

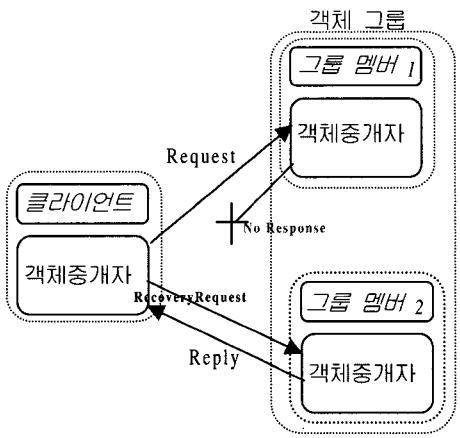
6.2 다중 호출 방지

객체 그룹 멤버들 중에 외부 객체에 대한 호출을 야기한 클라이언트가 접속된 그룹 멤버만이 외부 호출을 수행할 수 있게 제어함으로써 다중 호출을 방지한다. (그림 6)는 객체 그룹의 멤버가 외부 함수를 호출하는 과정을 나타낸다. 그림에서 그룹 멤버 2 만이 외부 호출을 수행할 수 있다. 다중 호출을 제어하기 위하여 본 논문에서는 쓰레드(thread)를 관리하는 방법을 사용한다. 해당 클라이언트가 접속되지 않은 노드에서는 멤버 함수를 호출하기 전에 객체 중개자에 외부 호출 금지를 알리기 위하여 자신의 쓰레드를 등록한다. 이 호출의 결과가 외부 객체를 호출할 때 객체 중개자는 등록된 쓰레드를 기반으로 외부 호출이 금지됨을 인지하고 다른 멤버로부터 호출 결과 값이 도착할 때 까지 대기케 한다.

함수 호출을 요구한 클라이언트가 접속된 멤버의 경우에도 함수 호출 전에 쓰레드를 등록한다. 이 때 등록된 쓰레드는 외부 함수 호출이 요구될 경우 결과 값을 다른 멤버 객체들에게 전송하는 것이 필요하다는 것을 의미한다. 이에 따라 함수 호출을 요구한 클라이언트가 접속된 멤버 객체의 경우에는 외부 호출을 수행하고, 결과 값을 다른 멤버들에게 전송한 후 리턴한다.

7. 연속 서비스 지원

서버 객체의 중복을 지원하는 것은 서버 객체가 수행되는 노드 및 이에 연결된 네트웍에 오류가 발생하더라도 클라이언트 요구는 중단됨이 없이 계속 진행하기 위함이다. 앞 장에서 기술한 바와 같이 본 논문에서는 각 클라이언트가 서버 객체 그룹의 특정 멤버에 접속하며, IIOP를 이용하여 함수를 호출하는 동작 구조를 지원한다. 이러한 동작 구조에서는 클라이언트가 접속되지 않은 멤버에서 발생하는 오류는 클라이언트의 수행에 영향을 미치지 않는다. 반면 클라이언트가 접속된 서버 객체에 문제가 발생하는 경우에 클라이언트의 수행에 영향을 미칠 수 있으므로, 이러한 경우에도 클라이언트의 요구가 정상적으로 수행됨을 보장하는 것이 필요하다.



(그림 7) 서버 객체 오류 시의 복구 메커니즘

본 논문에서는 클라이언트 응용이 서버 객체 그룹 멤버의 오류를 인지할 수 없도록 객체 중개자 레벨에서 복구를 지원한다. 즉 접속된 서버 객체 그룹의 멤버가 수행되는 노드나 네트웍에 고장이 발생하는 경우 객체 중개자는 동일 객체 그룹의 다른 멤버에 접속하여 클라이언트가 계속 수행할 수 있도록 한다. 이러한 방법을 사용하기 위해선 클라이언트와 서버 사이의 동작 과정을 나누어 판단하여야 한다. 클라이언트가 서버에 호출 메시지를 보내기 전이나 호출 메시지를 보내고 그 결과에 대한 응답을 받은 경우는 다른 멤버에 다시 접속함으로써 해결될 수 있다. 그러나 클라이언

트가 요청 메시지를 보내고 서버 객체가 이를 수행하는 도중에 문제가 발생하는 경우에 대해 고려하여야 한다. 이 경우에 메시지는 다른 멤버들에게 전송되어 수행되었을 수도 있고 다른 멤버에게 전송되지 못했을 경우도 있다. 따라서 이를 무시하고 다른 서버에 접속하여 해당 오퍼레이션을 다시 요청한다면 At-Most-Once Invocation을 지키지 못해 객체 그룹 멤버들 사이에 상태 불일치를 초래할 수 있다.

이를 해결하기 위하여 본 논문에서는 함수 호출을 요청한 클라이언트가 접속되지 않은 노드에서 수행되는 멤버들은 함수 호출을 수행한 결과 값을 해당 클라이언트로부터 다음 메시지가 도달할 때 까지 복구 큐에 저장한다. 클라이언트 객체 중개자는 요청 메시지를 보낸 후 응답 메시지를 받지 못했을 경우 다른 멤버에 접속하여 복구 메시지를 요청하여 응답을 받은 후 계속 진행 할 수 있게 된다. 이러한 과정은 클라이언트 응용에 투명하게 진행된다. (그림 7)은 호출 도중 서버 객체에 오류가 발생한 경우의 복구 과정을 나타내고 있다. 본 논문에서는 서버 객체의 복구를 지원하기 위한 메시지를 기존 IIOP에 추가한다. 추가된 메시지 타입은 *RecoveryRequest*이며, 메시지 구조는 Request와 동일하다. RecoveryRequest 메시지가 도착한 경우 객체 중개자에서는 복구 큐의 해당 호출의 결과가 존재하는 경우 결과 값을 돌려주고 그렇지 않은 경우 함수 호출을 수행한다.

8. 상호 운용성

본 장에서는 본 논문의 객체 그룹의 상호 연동을 두 가지 관점에서 점검한다. 기존의 CORBA 서비스를 활용하는 경우와 기존 CORBA 서비스를 객체 그룹 서비스로 확장하였을 때 기존 클라이언트가 본 논문의 개체 그룹 서비스를 활용할 수 있는 가를 점검한다.

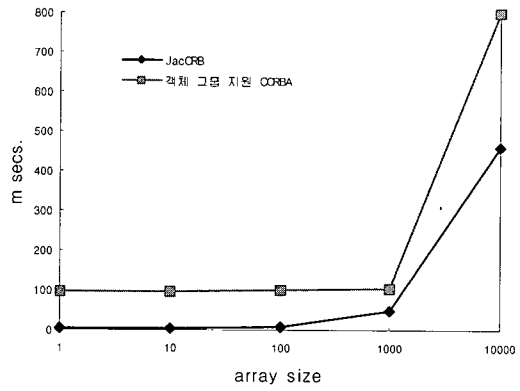
본 논문의 객체 그룹 구조에서는 클라이언트와 서버 객체 그룹 사이의 통신이 IIOP에 의해 이루어지기 때문에 기존 CORBA의 IOR 표현 형식을 그대로 사용하여 서버 객체에 접속한다. 따라서 기존 CORBA의 서비스를 사용하는 것에 전혀 문제가 발생하지 않는다.

기존 CORBA 서비스를 객체 그룹을 사용한 서비스로 확장하여 사용할 경우 기존 클라이언트가 해당 서비스를 이용할 수 있는 지를 점검한다. 클라이언트는 새로운 그룹 IOR을 이용하여 서버 객체 그룹 멤버 중 하나에 접속하고, 이 채널을 통해 IIOP 메시지를 보낸다. 본 논문에서도 IOR의 표현 형식, 접속 과정 그리고 클라이언트와의 통신을 위하여 IIOP 메시지를 사용하기 때문에 클라이언트에게는 객체 그룹에 관한 투명성을 제공한다. 단지 클라이언트가 접속된 객체 그룹 멤버에서 고장이 발생하였을 경우 클라이언트는 서버 객체가 중복되어 있음에도 불구하고 다른 멤버 객체에 접속할 수 없기 때문에 연속 서비스를 제공할 수 없다.

9. 성능 시험

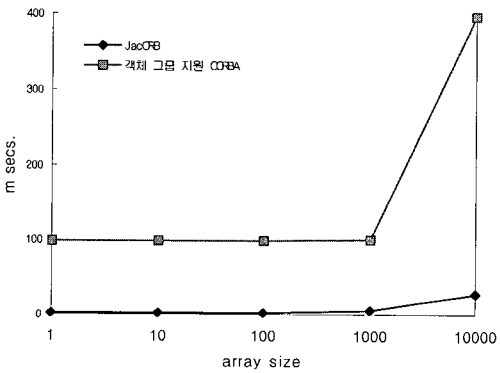
본 논문의 객체 그룹 지원 시스템은 설계상의 문제점을 점검하기 위한 시험 시제품으로 개발되었으며, 공개 소프트웨어들을 확장하는 방법으로 구현되었다. 객체 중개자로는 GNU의 JacORB 버전 0.9f1[3]을 사용하였으며, 다중 전송 프로토콜로는 MTP 프로토콜 규격을 구현한 MTP-2[1, 2]를 사용하였다. JacORB는 Java 언어로 구현된 반면, MTP-2는 C 언어로 구현되어 있기 때문에 이를 JacORB에 연결하기 위하여 Java 인터페이스를 지원하도록 확장하였다[19]. 본 장에서는 확장한 시제품의 성능을 JacORB와 비교하여 분석한다.

성능 시험을 위한 응용으로는 JacORB의 성능 시험에 사용되었던 벤치마크 프로그램[3, 4]의 서버 프로그램을 객체 중복을 허용할 수 있게 수정하였다. 벤치마크 프로그램의 서버 객체는 세 개의 함수를 제공한다. 하나는 매개변수 및 결과 값이 없는 단순 함수이며, 두 번째 함수는 Integer 배열을 매개 변수로 하며, 결과 값으로 매개 변수로 주어진 배열을 돌려주는 기능을 수행한다. 세 번째는 두 번째 함수와 유사하나 매개 변수 및 결과 값으로 Integer 변수 한 개를 멤버로 갖는 Struct 배열을 사용하는 함수이다. 즉 서버는 클라이언트의 함수 호출 메시지를 받은 후 매개 변수를 분리 한 후 이를 함수 결과 값으로 돌려주는 기능을 수행한다. 클라이언트에서는 배열 크기가 다른 매개 변수를 사용하여 서버 함수를 호출하고 결과 값이 돌아오는 동안의 시간을 측정한다. 첫 번째 함수를 사용한 측정 결과는 두 번째 함수를 사용하는 측정에서 배열의 크기를 1로 주었을 때와 동일하게 나타나기 때문에 본 장에서 결과를 별도로 나타내지 않았다.



(그림 8) Struct 배열을 사용한 성능비교

성능 측정은 Solaris 2.5.1.이 수행되는 3대의 SUN Ultra-1 과 Solaris 2.7이 수행되는 SUN Enterprise-450을 100Mbps 스위칭 허브로 연결된 환경에서 JDK1.1.7을 사용하여 시행하였다. JacORB 성능 측정은 SUN Ultra-1에 서버를, 그리고 Sun Enterprise-450에 클라이언트를 위치시킨 환경에서 시행하였다. 객체 그룹 지원 시스템은 SUN Ultra-1에 객체 그룹 멤버를 각 한 개씩 둠으로써 전체 3개의 중복 객체가 수행될 수 있도록 하고 클라이언트를 SUN Enterprise에 위치시켜 측정하였다. 실제 측정에서는 멤버의 수에 따른 결과 값을 구하였으나, 멤버 수의 변화에 따라 성능의 변화가 없었기 때문에 그림에 나타낼 수가 없었다. 시험은 200번을 수행시킨 결과 값을 평균하였다. (그림 8)은 Struct 배열을 사용하는 함수를 배열의 크기에 따라 측정한 결과를, (그림 9)는 integer 배열을 사용하는 함수의 측정 결과를 나타내었다.



(그림 9) Integer 배열을 사용한 성능 비교

(그림 8)과 (그림 9)는 유사한 크기의 변수를 사용함에도 불구하고 처리 속도의 차이가 큰 것을 보여준다. 이는 변수 값을 메시지로 변환하는 과정의 차이에서 발생한다. Integer와 같은 기본 타입의 경우에는 크기가 결정되어 있기 때문에 별도의 변환 과정이 필요치 않는 반면 확장 타입의 경우에는 변환 함수가 호출되어야 한다. 즉 (그림 9)는 클라이언트와 서버 사이에 메시지 전송에 소요되는 성능을 보여준 반면 (그림 8)은 데이터 변환에 소요되는 시간이 추가된 것이다. 실제 배열의 크기가 1000을 넘는 경우에는 JacORB에서 나타난 두 함수의 속도 차이는 객체 그룹 지원 시스템에서의 두 함수 수행 속도 차이와 거의 일치되게 나타남을 볼 수 있다. 이 같은 분석은 고장 감내 CORBA에서 성능은 거의 다중 전송 프로토콜의 성능에 좌우됨을 보여준다.

그러나 그림에 나타난 성능의 차이는 응용에서 소요되는 시간이 없는 상태이기 때문에 상대적으로 성능 저하 현상이 커 보일 수 있다. 실제 환경에서는 응용에서 수행되는 시간이 메시지 전달 시간에 비해 크기 때문에 성능 저하는 상대적으로 줄어 들 수 있다. 또 시험 결과는 현재의 구현 결과를 이용해서도 메시지의 크기가 4K 미만인 경우에는 활용 가능한 응용들이 있을 수 있음을 보여준다.

많은 응용에서 자료를 참조하는 경우가 많기 때문에 이러한 응용의 특성을 반영할 수 있는 메커니즘을 추가하는 것도 하나의 방법이 될 수 있다. 언어적 측면에서 이를 지원하는 것도 하나의 방법일 수 있으며[17], 서로 관련성이 없는 메시지들을 전체 순서화의 대상에서 제외함으로써 메시지 전송 속도를 향상시킬 수 있다.

10. 결론 및 향후 계획

본 논문에서는 CORBA에 객체 그룹을 쉽게 지원할 수 있는 동작 구조를 제안하고, 구현하였다. 구현된 객체 중계자는 서버 객체 중복의 투명성을 제공함으로써 기존 CORBA와 상호 운용될 수 있으며, 또 추가되는 응용 프로그래밍 인터페이스 함수를 최소화함으로써 기존의 서버 응용을 객체 그룹을 지원하는 서버 응용으로 쉽게 확장할 수 있음을 보였다. 그러나 성능 시험의 결과는 무엇보다 우선적으로 성능을 향상시키는 방법이 강구되어야 함을 나타내고 있다. 특히 효율적으로 동작하는 다중 전송 프로토콜을 개발하는 것이

성능을 향상시키는 지름길임을 보여주고 있다. 그러나 다중 전송 프로토콜의 성능을 향상시키는 것은 한계가 있다. 따라서 응용의 특성을 반영할 수 있는 방법을 제공함으로써 불필요한 성능 저하 요인을 제거할 수 있게 하는 것이 적은 노력으로 좋은 결과를 얻을 수 있는 방법으로 판단된다. 인터페이스 언어에 응용의 특성을 반영할 수 있는 방법을 추가하는 것은 향후 계획의 하나이다.

참 고 문 헌

[1] Armstrong S. et. al., "Multicast Transport Protocol," DARPA RFC 1301, 1992.

[2] Bormann, C., Ou, J. , Gehrcke. H-C., Kerschat, T. and Seifert, N., "MTP-2 : Towards Achieving the S.E.R.O. Properties for Multicast Transport," Technical Report of TU-Berlin 1994.

[3] Brose, G., "JacORB : Implementation and Design of a Java ORB," Procs. of DAIS'97, 1997.

[4] Brose, G., "JacORB Performance compared," http : //www.inf.fuberlin.de/~brose/jacorb/performance/results_09html, 1998.

[5] Ericsson, Iona Technologies and Nortel Networks, "Fault Tolerant CORBA," OMG Document orbos/98.10.10, 1998.

[6] Eternal Systems and Sun Micro Systems, "Fault Tolerance for CORBA Version 1.0-Initial Submission," OMG Document orbos/98.10.08, 1998.

[7] Felber, P., Garbinato, B., and Guerraoui, R., "A CORBA Object Group Service," Technical Report 97-223, Ecole Polytechque Fdrale de Lausanne, 1997.

[8] Highlander Communications et al., "Fault Tolerant CORBA using Entity Redundancy," OMG TC Document orobos/98.10.09, 1998.

[9] Maffeis, S., "Adding Group Communication and Fault-Tolerance to CORBA," USENIX, 1995.

[10] Miller, C. K., "Multicast Networking and Applications," Addison Wesley, 1999.

[11] Morgan, G., Shrivastava, S.K., Ezhilchelvan, P.D., Little, M.C., "Design and Implementation of a CORBA Fault-Tolerant Object Group Service," Technical Report of New Castle Unv., 1998.

[12] Narasimhan, P., Moser L. E. and Melliar-Smith P., "The Interception Approach to Reliable Distributed CORBA Objects," 3rd USENIX Conference on Object-Oriented Technologies and System, 1997.

[13] Objective Interface Systems, "Fault Tolerant CORBA Through Entity Redundancy," OMG TC Document orbos/98.10.03, 1998.

[14] OMG, "The Common Object Request Broker : Architecture and Specification," Revision 2.2, OMG, 1998.

[15] Oracle Corporation, "Fault Tolerance RFP," OMG TC Document orbos/98.10.13, 1998.

[16] Schneider, F. B., "The State Machine Approach," Lecture Notes in Computer Science, 1987.

[17] Shin, B. J. and Lee, T. H., "Design and Implementation of a Distributed Object Programming Language supporting Peer Replicated Object Model," Journal of KISS(C), KISS, 1999.

[18] Simon, B. and Spector, A., "Fault-Tolerant Distributed Computing," Lecture Notes in Computer Science 448, Springer-Verlag, 1990.

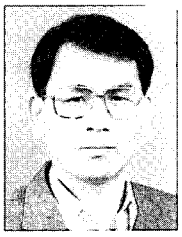
[19] SUN Micro System, "Java Native Interface Specification," SUN Document, 1997.

[20] Landis, S. and Maffeis, S., Building Reliable Distributed Systems with CORBA, Theory and Practice of Object Systems, Vol.3, No.1, John Wiley, April 1997.

[21] Haar, M., Cunninghan, R.and Cahill, V., Supporting CORBA Applications in a Mobile Environment, The Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, Seattle, Washington, August 15-20, 1999.

손 덕 주

e-mail : djson@etri.re.kr
1976년 서울대학교 수학교육과 졸업(학사)
1978년 한국과학기술원 전산학과 졸업(석사)
1978년~현재 한국전자통신연구원 책임연구원
관심분야 : 분산처리, 분산객체, 이동 컴퓨팅



신 범 주

e-mail : bjshin@etri.re.kr
1983년 경북대학교 전자공학과 (학사)
1991년 경북대학교 컴퓨터공학과 (석사)
1998년 경북대학교 컴퓨터공학과 (박사)
1999년~현재 한국전자통신연구원 책임연구원
관심분야 : 분산시스템, 트랜잭션처리, 이동컴퓨팅

남 궁 한

e-mail : nghan@etri.re.kr
1976년 고려대학교 물리학과 졸업
1982년 Columbia 대학교 전산학과 졸업
1982년~1986년 LG전자
1987년~현재 한국전자통신연구원 컴퓨터소프트웨어 연구소
관심분야 : 분산시스템, 이동컴퓨팅

진 성 일

e-mail : sijin@cs.chungnam.ac.kr
1978년 서울대학교 계산통계학과 (학사)
1980년 한국과학기술원 전산학과 (이학석사)
1994년 한국과학기술원 전산학과 (이학박사)
1983년~현재 충남대학교 컴퓨터과학과 교수
관심분야 : 데이터베이스, 멀티미디어시스템, 소프트웨어공학, 시뮬레이션 모델링 등