# 고장감내 CORBA를 지원하기 위한 객체중개자의 확장

## (An ORB Extension for support of Fault-Tolerant CORBA)

신 범 주 † 손 덕 주 †† 김 명 준 †††

(Bum-Joo Shin) (Duk-Joo Son) (Myung-Joon Kim)

**요 약** CORBA 응용에서는 서버 객체가 수행되는 노드 및 네트웍에 오류가 발생할 경우 전체 서비스가 중단되는 결과를 초래한다. 이 같은 문제를 해결하는 방법 중 하나는 동일한 서버 객체들을 여러 노드에 중복 수행시키는 것이다. 여러 노드에 중복되어 동일한 업무를 수행하는 객체들을 객체그룹이라 한다. 본 논문은 서버 객체의 고장감내를 지원하기 위하여 능동복제 방식의 객체그룹을 지원하는 고장감내 CORBA 모델을 제안하고 구현한 결과를 기술한다. 본 논문에서 제안하는 모델은 클라이언트와 서버 사이에 IIOP를 사용하기 때문에 기존의 CORBA 제품들과 상호 운용될 수 있을 뿐 아니라 추가되는 응용 프로그래밍 인터페이스를 최소화하는 장점을 제공한다. 또 응용의 특성에 따라 상태일치 과정을 피할 수 있게 하는 IDL구문을 제공함으로써 불필요한 성능저하를 방지할 수 있게 한다. 현재 능동 복제만을 지원하고 있지만 능동 복제를 지원하는 구조를 변경하지 않고 수동 복제 방식도 쉽게 지원할 수 있다.

***Abstract*** The failure of network and/or node on which server object is executed is a single point of system failure in the CORBA application. One of the possible ways to overcome such problem is to replicate server objects to several independent nodes. The replicated objects executing same tasks are called object group. In order to provide fault tolerance of server object, this paper proposes and implements new CORBA model that supports the object group based on active replication. The proposed model not only provides interoperability with existing CORBA application but also minimizes additional application interface required to support object group because it uses IIOP to exchange messages between client and server. And this paper extends IDL structure. Depending to application logic, it makes possible to prevent performance degradation caused by consistency maintenance. At present, this paper supports only active replication. But it can be easily extended to provide warm and/or cold passive replication without modification of architecture required for active replication.

## 1. 서 론

OMG CORBA[23]는 산업 표준으로 자리잡고 있는 분산 객체 시스템 규격이다. 그러나 클라이언트-서버 구조를 지원하기 때문에 CORBA 응용은 서버 노드 및 이에 연결된 네트웍에 고장이 발생할 경우 전체 시스템의 동작이 중단될 수 있는 단점을 갖는다. 이 같은 단점을 해결할 수 있는 가장 보편적이고 경제적인 방법중의 하나는 객체그룹을 지원하는 것이다[26]. 객체그룹은 동일한 업무를 수행하는 여러 객체가 동일한 상태를 유지함으로써 외부적으로 마치 하나의 객체처럼 동작하는 것을 의미한다.

기존의 CORBA를 객체그룹을 지원할 수 있도록 확장하는 방법은 가로채기 방식(Interceptor Approach)[21], 서비스 방식(Service Approach)[10][20] 그리고 객체중개자(ORB: Object Request Broker) 확장 방식(Integrated Approach)[18]이 가능하다. 가로채기 방식은 서버 객체에 보내는 객체중개자의 메시지를 시스템에 종속적인 방법으로 가로채어 객체그룹의 모든 멤버 객체들에게 메시지를 전달하는 방식이며, 서비스 방식은 이 같은 기능을 제공하는 서비스(COS: Common

† 정 회 원 : 한국전자통신연구원 자료저장 S/W팀 연구원
bjshin@etri.re.kr
†† 비 회 원 : 한국전자통신연구원 인터넷서비스연구부 연구원
djson@etri.re.kr
††† 종신회원 : 한국전자통신연구원 컴퓨터소프트웨어기술연구소 연구원
joonkim@etri.re.kr
논문접수 : 1999년 10월 20일
심사완료 : 2001년 2월 26일

Object Service)를 제공하고 클라이언트는 해당 서비스를 이용하여 객체그룹에 접속하는 방법이다. 반면 객체중개자 확장 방식은 객체중개자를 확장하여 객체그룹을 관리할 수 있도록 하는 방법이다.

이상의 세 가지 방법은 각각의 장단점을 가진다. 가로채기 방법은 클라이언트에게 객체그룹을 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 객체중개자의 내부 구조를 일부 변경하여야 한다는 점이 있으나 클라이언트 투명성을 제공할 수 있으며, 적절한 동작 구조를 제공함으로써 기존 응용들과 상호 호환성을 제공할 수 있다는 장점을 제공한다. 이 같은 장점을 얻기 위하여 본 논문은 객체중개자를 확장하는 방식을 이용하여 객체그룹을 지원한다.

객체그룹을 관리할 수 있도록 객체중개자를 확장함에 있어 고려되어야 할 문제는 객체그룹 멤버들의 상태를 일치시키기 위하여 사용되는 그룹통신 모듈을 접속하는 방법이다. 클라이언트와 서버 객체중개자에 각각 그룹통신 모듈을 접속하여, 클라이언트 객체중개자가 직접 그룹통신을 사용하여 서버 객체그룹을 호출하는 방법도 가능하다. 그러나 이 방법은 기존 CORBA와 호환되지 않는 단점이 있다. 본 논문은 이러한 단점을 해결하기 위하여 클라이언트가 그룹통신을 사용하지 않는 방법을 제안하고 구현한다.

본 논문의 다음 장은 CORBA에서의 객체그룹 지원에 관한 기존 연구를 분석하며, 3장은 본 논문에서 제안하는 모델을 기술한다. 4장에서는 제안 모델이 동작하기 위하여 제공되어야 할 모듈들에 대한 설계 내용을 기술한다. 4.1은 본 논문이 사용하는 객체그룹 관리 기법을, 4.2에서는 그룹의 개념을 지원하는 그룹RPC 설계에 대해 기술하며, 4.3은 서버 객체의 고장 발생 시에 이를 감지하고 응용에 투명하게 복구하기 위한 복구 모듈의 설계 내용을 기술한다. 5장에서는 구현 및 기능 검증 그리고 제안된 고장 감내 CORBA의 성능을 다른 고장 감내 CORBA 및 기존 CORBA와 비교하며, 6장에서 결론을 맺는다.

## 2. 기존연구

본 장은 객체그룹을 지원하는 CORBA에 대한 기존 연구 결과들을 비교 분석한다. 서비스 방식을 사용한 시스템으로는 OGS[10], New Castle 대학에서 개발한 시스템[20]이 있으며, 가로채기 방식을 이용하여 구현된 결과물[9][21] 그리고 Electra[18]및 Orbix+IsIs[14][15]와 같은 객체중개자 확장 방식을 사용한 것들도 있다. 서비스 방식 및 가로채기 방식에 대한 차이는 앞장에서 기술한 바 있기 때문에 본 장에서는 본 논문의 방식과 동일한 방식을 사용하는 Electra 및 Orbix+IsIs를 제안 모델과 비교한다.

Electra는 IsIs 및 Horus 등과 같은 다양한 그룹통신 모듈을 사용할 수 있는 유연한 구조를 제공하는 장점이 있는 반면 클라이언트 노드에 그룹통신 모듈이 제공되어야 객체그룹에 접속할 수 있기 때문에 기존 CORBA와의 상호 연동성을 제공할 수 없다는 단점이 있다.

Orbix+IsIs는 기존의 Orbix와 IsIs를 접목한 상용 제품이며, IsIs의 프로세스 그룹을 기반으로 고장감내를 지원한다. Orbix+IsIs는 응용의 특성에 따라 선택 가능한 세 가지의 통신 스타일을 제공한다. 클라이언트 응용의 요청을 객체그룹에 다중 전송하는 Multicast 스타일, 객체그룹의 특정 멤버에 접속할 수 있는 Client's Choice 스타일 그리고 프라이머리-백업과 유사한 동작을 지원하는 Coordinator/Cohort 스타일을 지원한다. Client's Choice 스타일은 본 논문의 모델과 유사하나 Orbix+IsIs에서는 읽기 전용 객체에 한해 동작 가능한 스타일이다. 따라서 Oribx+IsIs는 기본적으로 클라이언트 객체중개자가 다중전송을 통해 객체그룹에 접속하는 Electra와 동일한 모델을 사용한다. 이 같은 모델을 사용하는 경우 기존 CORBA 클라이언트 응용과의 상호 운용되지 않는 단점이 있다.

반면 본 논문에서 제안하는 모델은 클라이언트는 서버 객체그룹의 멤버들 중 하나와 IIOP(Internet Inter-ORB Protocol)를 사용해 통신하고, 메시지를 받은 서버 객체가 다른 멤버들에게 해당 메시지를 전송함으로써 객체그룹의 상태를 일치시키는 모델을 사용하기 때문에 기존 CORBA에서 수행되던 클라이언트와 상호 연동을 지원하는 장점이 있다.

OMG에서는 고장감내 CORBA의 표준을 제정하기 위한 작업을 진행하고 있다. 현재 5개의 제안서[8][9][12][22][24]가 제출되었으며, 제안서를 제출한 기관들 간에 단일 제안서를 만들기 위하여 절충하는 단계에 있다.

## 3. 제안모델

객체그룹을 구성하는 멤버 객체들의 상태를 효율적으로 일치시키기 위하여 다중전송 프로토콜의 사용이 필요하다[17][19]. 기존의 TCP/IP를 사용하던 객체중개자를 다중전송 프로토콜을 사용할 수 있도록 확장하는 방식은 두 가지가 가능하다. 첫 번째 구조는 그림 1에

나타낸 바와 같이 클라이언트의 객체중개자가 다중전송 프로토콜을 통해 함수 호출 메시지를 직접 객체그룹 멤버들에게 보내는 구조이며, 다른 한 가지는 논문에서 제안하는 그림 2의 구조이다. 본 논문은 클라이언트의 객체중개자가 객체그룹의 멤버들 중 하나에 TCP/IP를 통해 호출하고, 접속된 클라이언트로부터 함수 호출을 받은 서버 객체의 객체중개자가 다른 멤버 객체들에 다중전송 채널을 통해 전달하는 모델을 제안한다.

본 논문의 모델은 기존 모델에 비해 클라이언트가 접속된 네트웍 및 서버에 오류가 발생할 경우 추가적인 오류 복구 과정이 요구되는 단점을 가지는 반면 여러 장점들을 제공한다. 첫째, 클라이언트 객체중개자가 다중전송 프로토콜을 사용할 필요가 없기 때문에 기존 응용과 호환된다. 둘째, 다중전송 프로토콜로 인해 발생하는 클라이언트의 부하가 증가하지 않는다. 이 점은 제한된 자원을 가지는 노트북, PDA에도 클라이언트를 수행할 수 있기 때문에 향후의 응용 분야에 활용될 수 있다. 또 기존 제품을 쉽게 확장할 수 있을 뿐 아니라 프라이머리-백업 방식도 쉽게 구현할 수 있다는 점도 이 구조가 갖는 장점들이다. 또 이 같은 모델을 사용함으로써 객체의 상태에 영향을 미치지 않는 오퍼레이션의 경우 다중전송 프로토콜을 사용치 않게 함으로써 응용의 특성에 따라 성능 저하를 최소화하는 것이 가능하다. 본 논문에서는 IDL 구문을 확장하여 이 같은 기능을 지원한다.

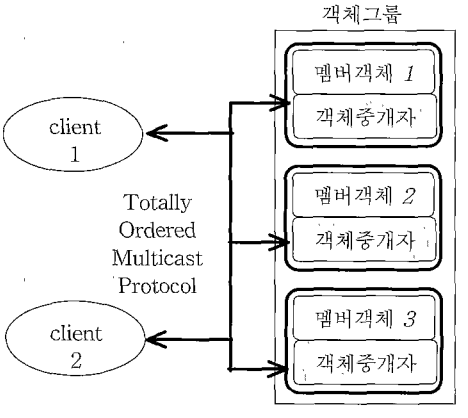


그림 1 객체그룹 지원을 위한 기존 모델

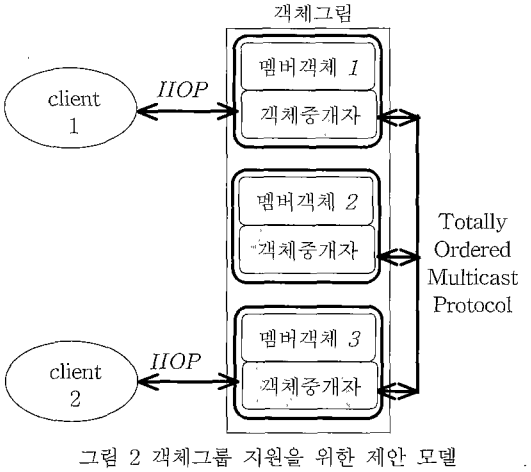


그림 2 객체그룹 지원을 위한 제안 모델

본 장의 서두에서 기술한 바와 같이 본 논문의 모델을 사용할 경우 객체중개자가 클라이언트와 서버 사이에 발생하는 고장을 인지하고 복구하는 것이 필요하다. 이를 위하여 본 논문에서 적용하는 고장 모델은 Fail-Stop 모델이다. Fail-Stop 모델은 다른 복잡한 모델들과 달리 각 컴퓨팅 요소에 고장이 발생한 경우, 해당 요소가 아무런 동작을 하지 않는 모델이다[26]. 이 같은 모델에서는 네트웍의 단절, 노드의 고장 및 프로세스의 고장을 구별할 수 없으며, 고장을 감지하기 위하여 일정 시간마다 메시지를 교환하는 기능이 필요하다. 본 논문에서는 고장 감지를 위하여 하부의 통신 모듈이 지원하는 것과 별도로 객체중개자 수준에서 일정 시간마다 keep-alive 메시지를 교환하는 고장인지 방법을 사용한다.

고장감내 기능은 서버 객체에 국한되며 클라이언트 노드의 고장 그리고 클라이언트 프로세스의 고장은 본 논문의 고장감내 밖의 범위이므로 지원되지 않는다. 반면 클라이언트에 접속된 네트웍의 고장은 서버의 고장으로 인정되기 때문에 감내 기능이 지원되며, 특정 프로세스에 발생한 고장은 해당 프로세스에 의해 지원되는 모든 객체의 고장으로 인정한다.

## 4. FT-CORBA 설계

CORBA의 동작 모델은 RPC(Remote Procedure Call)를 기반으로 한다. 서버 그룹의 개념을 지원할 수 있도록 확장한 RPC를 그룹RPC라 한다[18]. 그룹RPC는 RPC의 개념을 유지하면서 3장에서 기술하였던 그룹 접속 모델을 감출 수 있어야 할 뿐 아니라 고장을 감내할 수 있어야 한다. 본 장은 제안 모델을 기반으로 그룹RPC를 지원하기 위하여 추가되어야 하는 기능들의 설계 내용을 기술한다.

### 4.1 그룹관리

클라이언트가 그룹RPC를 호출할 경우 함수 호출이 서버 객체그룹을 구성하는 모든 멤버들에게 전달되어야 한다. 이 같은 동작은 객체그룹을 구분할 수 있는 메커

니즘이 제공되고, 객체그룹을 구성하는 멤버들에 대한 정보가 관리될 때 가능할 수 있다. 객체그룹을 구분하는 메카니즘은 객체그룹을 구성하는 멤버 객체를 프로세스에 사상하는 방법에 따라 달라질 수 있다. 멤버 객체를 프로세스에 사상하는 방법으로 두 가지 모델을 고려할 수 있다. 첫 번째 모델은 프로세스 그룹과 객체그룹을 일치시키는 방법이다. 즉 동일한 프로그램을 여러 노드에 수행케 하는 모델이다. 두 번째 모델은 그림 3과 같이 프로세스에 독립적인 객체그룹을 지원하는 모델이다. 즉 서로 다른 업무를 수행하는 프로세스들 사이에 객체그룹을 구성하는 모델이다.

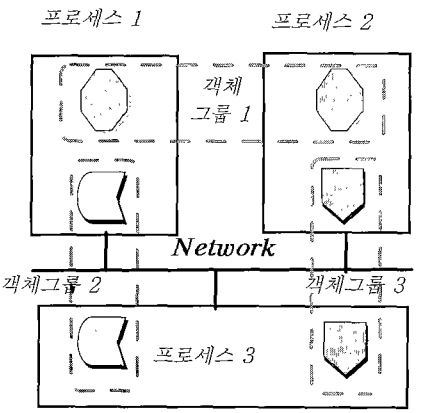


그림 3 객체그룹 구성 모델

본 논문은 보다 유연한 객체그룹을 지원하기 위하여 프로세스에 독립적인 모델을 지원한다. 이 같은 모델을 하나의 IP 주소를 공유하는 다중전송 프로토콜에 맵핑하기 위해서는 다중전송 채널에 도착한 메시지들 중에 자신이 관리하는 멤버 객체와 관련된 메시지를 발췌하는 기능을 제공하여야 한다. 이를 위하여 본 논문은 다중전송 메시지가 객체그룹의 이름을 포함하며, 그룹 이름을 기반으로 메시지를 발췌하는 독립 쓰레드를 두는 구조로 설계되었다. 이 같은 구조에서 적절한 메시지를 발췌할 수 있도록 객체중개자가 자신이 제공하는 멤버 객체와 객체그룹의 정보를 관리한다.

4.1.1 멤버관리

서버 프로그램이 객체(프로그래밍 언어에 종속된 객체)를 생성한 후, 이를 CORBA 객체로 만들기 위해서는 해당 객체를 객체중개자에 등록하여야 한다. 본 논문의 객체그룹 멤버도 유사한 방법으로 객체중개자에 등록하도록 설계되었다. 본 논문은 객체그룹을 생성하기 위하여 명시적인 인터페이스를 제공하지 않으며, 객체그룹의 멤버로 등록할 때 그룹의 이름이 파라미터로 주어지기 때문에 해당 그룹이 존재하지 않을 경우 생성하는 방법을 사용한다. 이를 위하여 기존의 ***obj_is_ready*** 인터페이스 함수[23]와 유사한 ***obj_group_is_ready*** 인터페이스 함수를 제공한다. 이 함수를 호출할 때 그룹 이름과 객체 참조자가 매개 변수로 주어진다. 객체는 이 함수를 통해 동적으로 객체그룹에 가입할 수 있다. 함수의 매개변수로 제공되는 그룹 이름에 의해 객체그룹이 구분되며, 모든 노드의 객체중개자는 자신과 관련된 그룹 및 멤버 정보를 관리한다. 새로운 멤버가 가입될 때 기존 멤버에 대한 정보 및 객체의 상태 정보는 마스트 멤버가 존재하는 객체중개자에 의해 전송된다. 마스트 멤버는 최초로 멤버에 가입한 멤버로 결정되며, 마스트 멤버가 존재하는 노드에 오류가 발생할 경우 가입 우선순위에 의해 다음 마스트 멤버가 결정된다. 새로운 멤버가 추가될 경우 마스트 멤버가 존재하는 노드의 객체중개자는 기존 멤버들로부터 준비 완료 메시지를 받은 후 멤버 가입을 요청한 노드에 멤버 정보 및 상태 정보를 전송한다.

4.1.2 그룹참조

CORBA는 클라이언트가 서버 객체에 접속 시 사용되는 IOR(Interoperable Object Reference) 이라는 객체 참조 구조를 정의하며, IIOP는 IOR에 포함되는 정보에 대하여 정의한다. 즉 IIOP에서는 IOR이 노드, 포트 그리고 서버 객체 식별자에 대한 정보를 제공하도록 정의하고 있다[23]. IOR이 원격객체를 표현하는 것과 같이 객체그룹에 접속하기 위해서는 객체그룹의 멤버 객체에 관한 정보를 포함하는 객체그룹 참조 구조가 필요하다.

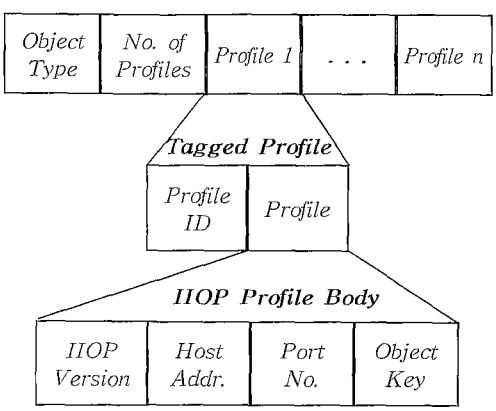


그림 4 객체그룹 참조구조

본 논문은 기존 CORBA와의 상호 운용을 지원하기 위하여 IOR을 변경하지 않고, 단지 의미를 확대하여 사용하도록 설계되었다.

객체그룹을 표현하는 객체 참조 구조는 여러 개의 프로화일을 가지며, 각 프로파일은 멤버객체에 관한 정보를 표현한다. 객체그룹 참조 구조를 기존의 구조와 구분하기 위하여 객체 타입을 표현하는 필드에 특정 표시어를 추가한다. 그림 4는 본 논문의 객체그룹 참조 구조를 나타낸다. IIOP는 객체 참조 구조가 적어도 하나의 프로화일을 가져야 한다고 정의되어 있기 때문에 이 구조는 CORBA IIOP에서 정의하는 객체 참조 구조와 동일한 것으로 인정될 수 있다.

클라이언트가 그림 4의 객체그룹 참조 구조를 사용하여 함수 호출을 할 때 서버 객체그룹멤버들 중 하나의 멤버 객체를 선정하여 접속하여야 한다. 기존 CORBA와 같이 첫 번째 프로화일의 객체에 접속할 경우 첫 번째 멤버에만 클라이언트 접속이 집중될 수 있다. 이를 방지하기 위하여 본 논문은 객체그룹 참조 구조에 표현된 프로화일들 중 무작위로 하나를 선정하여 접속한다. 그러나 이러한 방법도 서버의 완전한 부하 분산을 실현하기는 불충분하다. 따라서 서버의 부하 정도에 따라 접속할 서버를 선택하는 부하 분산 메커니즘이 추가되는 것이 바람직하다.

4.1.3 상태전달

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

본 논문은 함수 호출 메시지를 제어하기 위하여 상태전달 과정 동안 호출 메시지를 저장하는 버퍼를 사용하며, 객체의 상태를 메시지에 저장하고 복구하기 위한 방법으로 Java에서 제공하는 Serializable 인터페이스를 이용한다. 본 논문의 객체그룹을 사용하는 모든 객체는 Java Serializable 인터페이스를 상속하기 때문에 객체중개자에서는 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.

### 4.2 호출제어

메시지의 전송과 달리 RPC에서는 함수 호출 메시지에 대한 응답 메시지가 반드시 존재한다. 서버 객체그룹을 구성하는 모든 멤버들이 함수 호출에 반응하여 응답을 보내게 될 경우 클라이언트는 중복된 여러 개의 응답을 받는다. 이 같은 문제를 다중 응답이라 한다. 또 서버 객체의 멤버 함수가 수행 중에 클라이언트 자격으로 다른 서버 객체의 멤버 함수를 호출하는 경우, 모든 멤버들이 호출 메시지를 보내게 되면 해당 서버 객체는 멤버 수만큼의 중복된 함수 호출 메시지를 받게 된다. 이를 다중 호출이라 한다. 그룹 RPC를 지원하기 위해서는 이 같은 문제들이 해결되어야 한다.

본 논문은 다중 응답 및 다중 호출을 방지하기 위하여 매우 단순한 방법을 사용한다. 클라이언트가 접속된 서버 객체그룹의 멤버만이 클라이언트에게 응답 메시지를 보낼 수 있을 뿐 아니라 외부 서버 객체에 함수 호출 메시지를 보낼 수 있게 제어함으로써 그룹 RPC를 지원한다. 그림 5는 세 개의 멤버로 구성된 객체그룹이 수행 중 외부 객체를 호출하는 과정을 나타낸다. 클라이언트가 접속된 멤버 2 만이 외부 객체를 호출하고, 결과값을 클라이언트에게 보내는 구조를 보이고 있다. 클라이언트는 객체그룹의 불특정 멤버 객체들에게 접속할 수 있다. 단지 외부 함수 호출을 야기시키는 함수를 호출한 클라이언트가 접속된 멤버 객체만이 외부 함수를 호출할 수 있도록 제어한다. 동작 과정에서 요구되는 제어 방법은 다음 절에서 다룬다.

4.2.1 다중응답 제어

본 논문이 제안하는 모델에서는 다중 응답 방지가 쉽게 해결될 수 있다. 객체그룹의 멤버 객체는 자신에게 접속된 클라이언트가 보낸 함수 호출 메시지에 대해 응

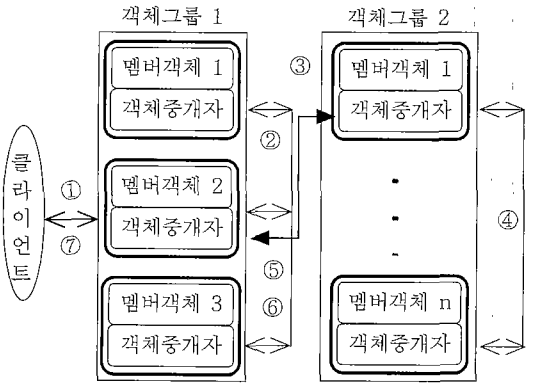


그림 5 객체그룹의 외부 객체 호출 과정

답하며, 자신에게 접속되지 않은 클라이언트의 요청에 대해서는 응답 메시지를 보내지 않도록 제어된다.

그림 5에 나타낸 바와 같이 클라이언트가 접속되지 않은 객체그룹의 다른 멤버(그림 5의 객체그룹 1의 멤버 1 과 멤버 3)들은 클라이언트와의 세션을 갖고 있지 않기 때문에 결과 값을 보낼 수 있는 방법이 없다. 다중 응답을 제어하기 위하여 본 논문은 IIOP에 정의된 메시지에 클라이언트가 접속된 멤버 객체에 관한 정보를 추가한 다중 전송 메시지를 정의한다. 클라이언트 객체중개자로부터 TCP 채널을 통해 함수 호출 메시지를 받은 멤버 객체의 객체중개자는 함수 호출 메시지 정보에 자신과 관련된 정보를 추가한 후 다중 전송 채널을 통해 그룹 멤버들에게 보낸다. 그룹 멤버들은 다중 전송 채널에 도달한 메시지의 정보를 바탕으로 클라이언트에 응답 메시지를 전송할 것인지를 결정한다

4.2.2 다중호출 제어

다중 호출을 제어하기 위하여 본 논문은 쓰레드를 관리하는 방법을 사용한다. 멤버 함수 호출을 요구한 클라이언트가 접속된 서버 노드(그림 5의 객체그룹 1의 멤버 객체2)는 외부 함수 호출이 허가되는 반면 그렇지 않은 노드(그림 5의 객체그룹 1의 멤버 객체1과 3)는 외부 함수 호출이 불가능하도록 제어된다. 이 같은 제어를 위하여 객체중개자는 두 개의 리스트를 관리한다. 외부함수 호출금지 리스트와 외부함수 호출허가 리스트가 그것이다. 이 외에 다중 전송 채널에 도착한 함수 호출 메시지를 저장하기 위한 큐 그리고 각 객체 테이블에는 외부 함수 호출이 금지될 경우 블록하기 위한 자료구조를 관리한다. 이상의 자료구조를 기반으로 독립적인 쓰레드로 동작하는 다중전송채널 관리기와 서버객체함수 호출기에 의해 다중 함수 호출이 제어된다.

3장에서 기술한 바와 같이 객체그룹의 멤버 객체를 제공하는 노드에서는 TCP 채널을 통해 클라이언트의 함수 호출 메시지를 받는다. 이 때 함수 호출 메시지를 직접 수행하지 않고 다중전송 채널에 송신함으로써 다른 멤버들도 함수 호출 메시지를 수행할 수 있게 한다. 다중전송 채널에 보낸 메시지는 다른 노드들과 마찬가지로 자신에게도 순서대로 도착한다. 다중전송 채널에 수신된 메시지는 해당 메시지를 송신한 노드의 정보를 제공한다. 다중 함수 호출은 메시지에 포함된 송신 노드에 관한 정보를 바탕으로 다음의 그림에 기술되는 방법을 사용하여 제어한다. 그림 6은 다중전송채널 관리기의 동작과정을 그림 7은 서버객체함수 호출기의 동작과정을, 그리고 그림 8은 외부함수 호출을 위하여 객체중개자에서 지원하는 함수의 기능을 가상 코드로 나타내었다.

다중전송 채널에 도착한 호출 메시지는 다중전송 채널 관리기에 의해 함수 호출 큐에 놓인다(그림 6의 라인4~6). 서버객체 함수 호출기는 큐에 놓인 메시지에 포함된 송신 노드 정보를 이용하여 외부 함수 호출이 가능한 가를 판단한 후 자신의 쓰레드 번호를 해당 리스트에 등록한 후 서버 객체 함수를 호출한다(그림 7의 라인3~9). 이때 호출된 서버 객체 함수가 수행 중에 클라이언트 자격으로 다른 서버 객체 함수를 호출하는 경우가 발생할 수 있다(그림 5의 과정 3 참조). 외부 객체 함수의 호출은 스텁 파일(stub file)을 통해 호출하므로 그림 7에 나타날 수 없으나 함수가 호출됨을 보이기 위하여 그림 7의 라인 10~12에 나타내었다. 외부 객체 함수를 호출할 경우 스텁 파일을 통해 객체중개자의 invoke()가 호출된다. 따라서 invoke()는 서버객체함수 호출기의 쓰레드를 통해 호출된다.

```
1 while(true){
2  fetch a message from multicast channel;
3  switch(message type){
4  case REQUEST:
5     enqueue this message to queue;
6     break;
7  case RESULT_VALUE:
8     attach this message to object table;
9     wake up the thread;
10    break;
11 . . .
12  }
13 }
```

그림 6 다중전송채널 관리기의 동작

```
1 while(true){
2  fetch a message from request queue;
3  if(message is sent from other node ){
4     register this thread with outgoing-call disable list;
5  }
6  else{
7     register this thread with outgoing-call enable list;
8  }
9  call function of server object;
10 if(the function of server object requires outgoing call){
11    orb.invoke();
12 }
13 if(message is sent from other node ){
14    save result value for recovery;
15    remove thread from outgoing-call enable list;
16 }
17 else{
18    remove thread from outgoing-call disable list;
19    send reply message to client;
20 }
21 }
```

그림 7 서버객체 함수 호출기의 동작

```
1 invoke(){
2 if(this thread is registered with outgoing call disable
  list){
3  block this thread till result value is arrived;
4  }
5  else{
6  send request message to remote object;
7  if(this thread is registered with outgoing call enable
  list){
8        multicast result to other members;
9    }
10 }
11 return;
12 }
```

그림 8 외부함수 호출을 위한 객체중개자의 함수

호출된 invoke()는 외부함수 호출금지 리스트에 자신의 쓰레드가 등재되지 않은 경우에 한해 외부 함수 호출 메시지를 보낸다(그림 8의 2~6). 또한 객체중개자는 외부 서버 객체로부터 응답 메시지를 받았을 때 자신의 쓰레드가 외부함수 호출허가 리스트에 등재되었을 경우 다른 멤버들에게 함수 호출 결과 메시지를 보낸다(그림 8의 7~11). 다중전송 채널에 외부 함수 호출 결과 값이 도착하면 다중전송 채널관리기가 대기 상태(그림 8의 3)인 쓰레드를 깨운다(그림 6의 7~10). 깨어난 쓰레드는 결과 값을 가지고 리턴한다(그림 8의 11). 함수호출허가 리스트에 등록한 것은 결과 값을 다른 멤버 객체들에게 전송해야 함을 의미하며 객체그룹을 지원하지 않는 객체 호출의 경우와 구분하기 위한 것이다. 결과 값을 복구 큐에 저장하는 것은 서버 객체에 오류가 발생할 경우 클라이언트 응용에 투명하게 복구하기 위한 것이다. 이와 관련된 설계 내용은 다음 절에서 자세히 기술한다.

### 4.3 고장 복구

본 논문에서는 서버에 오류가 발생하는 경우 클라이언트의 객체중개자가 정상적으로 동작하는 서버 그룹 멤버에 재 접속하여 서비스가 진행될 수 있게 한다. 객체중개자에서 이루어지는 재접속 과정은 클라이언트 응용에게는 감추어진다.

클라이언트 응용에게 서버의 고장에 대한 투명성을 제공하기 위하여 본 논문은 기존의 IIOP 메시지에 ***RecoveryRequest*** 메시지를 추가한다. 추가된 메시지 구조는 Request 메시지와 동일하다. 클라이언트가 접속된 서버에 고장이 발생하였을 경우 클라이언트의 객체중개자는 객체그룹 참조구조(그림 4)에서 서버 객체 그룹 멤버를 한 개 선정한 후 RecoveryRequest 메시지를 보낸다. 이 메시지를 받은 서버 객체중개자는 이를 수행하여야 할지 또는 수행치 않고 결과 값을 보내어야 할지를 결정하여야 한다. 이 같은 결정은 고장 발생 시점에 따라 달라진다. 클라이언트 객체중개자가 서버에 호출 메시지를 보내기 전에 고장이 발생한 경우 서버는 RecoveryRequest를 수행하면 된다. 그러나 클라이언트가 호출 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 다시 수행한다면 At-Most-Once Invocation을 지키지 못해 객체그룹 멤버들 사이에 상태 불일치를 초래할 수 있다.

이를 해결하기 위하여 함수 호출을 요청한 클라이언트가 접속되지 않은 객체그룹 멤버들은 함수 호출을 수행한 결과 값 및 함수 호출 번호를 해당 클라이언트로부터 다음 메시지가 도달할 때까지 저장한다. 클라이언트 객체중개자가 함수 호출 메시지를 보낸 후 서버의 고장으로 인해 응답 메시지를 받지 못했을 경우 다른 서버 객체그룹의 멤버에 접속하여 RecoveryRequest 메시지를 전송한다. 해당 서버 객체는 새로 도착한 메시지의 함수호출 번호를 저장된 자료의 함수호출번호와 비교하여 수행을 할 것인지 혹은 결과 값만 돌려줄 것인지를 판단하게 된다. 그림 9는 이러한 경우의 복구 과정을 나타내고 있다.

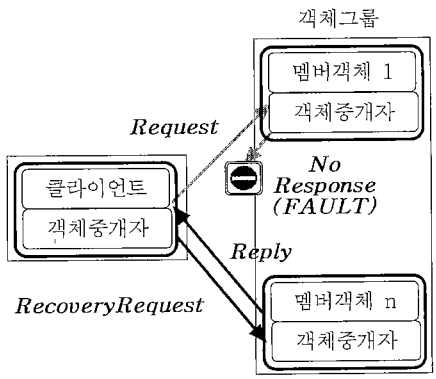


그림 9 고장 복구 과정

### 4.4 IDL 확장

객체그룹의 상태를 일치시키기 위하여 요구되는 동작 과정들은 시스템의 성능에 많은 영향을 미친다. 따라서 객체그룹의 상태에 영향을 미치지 않는 함수 호출의 경우 상태 일치 과정을 생략할 수 있기 때문에 성능을 향

상시킬 수 있다. 성능을 향상시키기 위해 프로그래밍 언어에서 이 같은 응용의 특성을 반영하기 위한 방법들이 제안되고 있다[29]. 응용의 특성을 반영하여 성능을 향상시킬 수 있도록 하기 위하여 본 논문에서는 그림 10과 같이 IDL을 확장한다.

*(88) <op_attribute> ::= oneway*
*| **readonly***

그림 10 IDL 문법의 확장

그림 10의 IDL 문법은 OMG의 문서 CORBA/IIOP 2.3[23]에서 관련된 문법만을 발췌한 것이다. 추가된 구문은 **readonly**이다. 확장된 문법은 객체의 멤버 함수가 readonly로 선언될 수 있음을 나타내며, readonly로 선언된 멤버 함수는 해당 객체의 내부 상태를 변경하지 않는다는 것을 의미한다. 클라이언트가 직접 다중전송을 사용하는 경우는 이러한 구문의 특성을 반영하기 힘든 반면 본 논문의 모델에서는 쉽게 적용할 수 있다. 그림 11은 추가된 IDL 문법을 이용하여 기술된 인터페이스 정의 예를 보이고 있다.

```
interface account{
    long        debit(in long value);
    long        credit(in long value);
    readonly    long   amount();
};
```

그림 11 확장 문법을 사용 예

## 5. 구현 및 성능분석

### 5.1 구현환경

본 논문은 기존의 객체중개자를 확장하여 다중전송 프로토콜을 결합하는 방법으로 구현하였다. GNU의 JacORB 버전 0.9f1[4]를 기본 객체중개자로 사용하였으며, 다중전송 프로토콜로는 MTP 프로토콜 규격을 구현한 MTP-2[1][3]를 사용하였다. JacORB는 Java 언어로 구현된 반면, MTP-2는 C 언어로 구현되어 있기 때문에 이를 JacORB에 연결하기 위하여 Java Native Interface[28]를 사용하여 Java용 클래스를 추가하였다. 구현된 FT-CORBA상에서 JacORB의 예제 프로그램들을 대상으로 서버 부분을 일부 수정하여 수행시킨 결과 정상적으로 수행됨을 볼 수 있었다. MTP-2를 사용한 버전과는 별도로 하드웨어에 독립적으로 수행할 수 있도록 Java로 구현된 다중 전송 소프트웨어인 Java Groups[2]를 사용한 버전도 지원한다. 본 장은 구현 결과물에 대하여 기존 CORBA와의 상호운용성과 고장감내를 검증하고 성능을 분석한다.

### 5.2 기능검증

5.2.1 상호운용

상호운용은 두 가지 측면에서 고려되어야 한다. 기존 CORBA 클라이언트와 FT-CORBA 서버 사이의 상호운용 및 기존 CORBA 서버와 FT-CORBA 클라이언트와의 상호운용의 두 가지이다. 본 논문은 서버 객체의 고장감내를 지원하기 때문에 후자의 경우는 고려할 가치가 없다. 반면 전자의 경우는 지원할 필요가 있다. 본 논문은 클라이언트와 서버의 연결을 기존 CORBA와 동일한 IIOP를 사용하기 때문에 기존 클라이언트가 서버 객체의 접속 및 호출에 아무런 문제가 없다. 단지 클라이언트가 접속된 객체그룹 멤버에서 오류가 발생하였을 경우 클라이언트는 서버 객체가 중복되어 있음에도 불구하고 다른 멤버 객체에 접속할 수 있는 기능을 제공치 않기 때문에 고장을 감내할 수 없다. 기존 CORBA 클라이언트 응용을 FT-CORBA의 서버에 접속 운용됨 확인하였다.

5.2.2 고장감내

고장감내의 검증을 위하여 서버 프로세스들 중 하나를 강제로 죽이는 방법과 서버 프로세스가 수행되는 노드의 네트웍을 강제로 단절시키는 방법을 사용하였다. 서버 프로세스를 강제로 죽였을 때 클라이언트의 TCP는 고장을 즉각적으로 파악하였으며, 객체그룹을 구성하는 다른 서버에서도 해당 프로세스의 고장을 인지하였다. 반면 네트웍을 단절시킨 경우는 객체그룹을 구성하는 다른 노드에서는 해당 노드의 고장을 인지하는 반면 클라이언트 TCP는 해당 서버의 고장을 인지하지 못하였다. 네트웍 고장을 인지하기 위하여 본 논문에서는 객체중개자 수준에서 서버 노드를 점검하는 방법을 사용한다. 클라이언트 객체중개자는 일정 시간마다 keepalive 메시지를 전송하고, 서버 객체중개자는 이에 반응하는 방법이다. 이 방법은 프로세스 수준의 고장인지 모델을 사용하는 경우에는 적절하게 동작하지만 객체 수준의 고장인지 모델을 사용하기 위해서는 수정되어야 할 부분이다.

### 5.3 성능분석

객체그룹을 통한 고장감내는 객체그룹 맴버의 상태를 일치되게 유지하기 위한 추가 오퍼레이션으로 인해 성능이 저하된다. 성능 저하의 정도 및 기존 고장감내 CORBA와의 차이를 확인하기 위해 JacORB의 벤치마크 프로그램[5]을 사용하여 성능을 측정하고, 이를 JacORB 및 OGS와 비교하였다.

성능 측정을 위해 기존 벤치마크 프로그램이 객체그룹을 지원하는 응용으로 동작할 수 있도록 하기 위하여 객체를 객체중개자에 등록하는 부분을 수정하였다. 또한 OGS의 성능측정은 클라이언트가 GroupAccess 객체에 접속하도록 수정하였으며, Untyped 서버로 동작하도록 서버 프로그램을 수정하였다.

성능 측정은 Solaris 2.5.1이 수행되는 3대의 SUN Ultra-1과 Solaris 2.7이 수행되는 SUN Enterprise-450 1대를 100Mbps 스위칭 허브로 연결된 환경에서 JDK1.1.7을 사용하여 시행하였다. JacORB 성능은 SUN Ultra-1에 서버를, 그리고 Sun Enterprise 450에 클라이언트를 위치시킨 환경에서 측정하였으며, FT-CORBA 및 OGS는 SUN Ultra-1에 객체그룹 멤버를 각 한 개씩 둠으로써 전체 3개의 멤버로 구성된 객체그룹이 수행될 수 있도록 하고 클라이언트를 SUN Enterprise에 위치시켜 측정하였다. 시험은 200 번을 수행시킨 결과 값을 평균하였다.

벤치마크 프로그램은 struct 배열을 매개 변수로 하는 함수와 integer 배열을 매개 변수로 하는 함수로 구성된다. 각각의 함수에서 매개 변수로 주어지는 배열의 크기에 따라 측정한 결과를 그림 12, 그림 13에 나타내었으며, 그림 14는 integer 배열을 사용하는 함수를 본 논문에서 제안한 IDL 구문인 readonly로 정의하였을 때의 객체그룹 멤버를 3개 갖는 환경에서 FT-CORBA 성능을 JacORB와 OGS의 객체그룹 멤버를 1로 했을 때 성능과 비교한 것이다.

그림 12과 그림 13에 나타낸 성능 측정 결과는 객체그룹 지원에 따른 성능 저하가 매우 크다는 것을 보여주고 있다. 물론 벤치마크 프로그램의 내용이 통신에 소요되는 시간이 부각되는 내용으로 작성되어 있기 때문에 성능 저하가 상대적으로 심각해 보일 수 있다. 그림 12와 그림 13에 나타난 성능 저하 율의 차이는 이러한 점에 기인한다. 그러나 실제 환경에서는 객체에서의 수행 시간이 통신에 소요되는 시간 보다 크기 때문에 성능 저하율이 줄어든다. 실제 환경을 고려하더라도 응용의 특성에 따라 불필요한 상태 일치 과정을 피할 수 있도록 함으로써 객체그룹 지원에 따른 성능 저하를 최소화하는 것은 필수적이다. 그림 14에서 볼 수 있듯이 본 논문에서 제안하는 readonly 구문은 응용의 특성에 따라 최적의 성능을 지원할 수 있는 기반을 제공한다.

또한 성능 측정 결과는 객체그룹을 지원하는 시스템에서는 다중전송 프로토콜의 특성이 전체 시스템의 성능에 가장 큰 영향을 미친다는 것을 나타낸다. 그림 12와 그림 13에 나타난 FT-CORBA/MTP-2와 FT-CORBA/JavaGroups의 측정 결과와 같이 동일한 시스템 구조에서도 하부의 다중 전송 프로토콜에 따라 성능이 매우 상이할 수 있다. 일반적으로 다중전송 프로토콜은 그룹의 크기, 전송 데이타의 크기에 따라 성능이 달라질 수 있기 때문에 응용의 특성에 맞는 프로토콜의 선정도 중요하다.

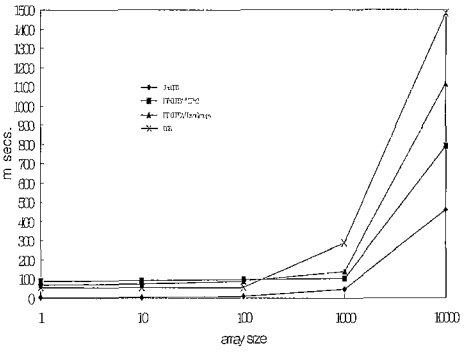


그림 12 Struct 배열을 사용한 성능비교

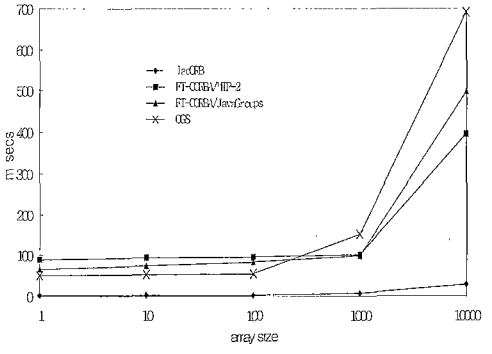


그림 13 Integer배열을 사용한 성능비교

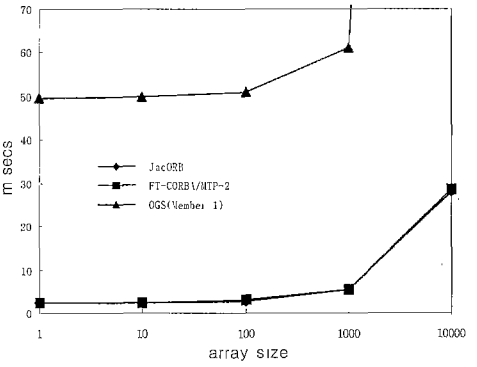


그림 14 readonly 함수의 성능 비교

본 연구와 관련하여 보다 우수한 성능을 제공하는 다중 전송 프로토콜의 개발을 진행하고 있다.

응용 수준의 성능 비교와는 달리 고장감내를 지원하기 위하여 서버 객체중개자의 주요 동작 요소들에서 수행되는 시간을 알아보는 것도 나름의 의미가 있다. 이를 위하여 서버 객체중개자가 TCP 채널로부터 클라이언트 메시지를 받은 후와 해당 메시지를 변환하여 다중 전송 채널에 전송하기 전, 다중 전송 채널에 도착한 메시지를 수신한 후, 그리고 수신한 메시지를 매개 변수로 변환하고 해당 함수를 호출한 후 클라이언트에 응답 메시지를 보내기 전에 시간을 측정하였으며, 결과는 표 1과 같다.

표 1 동작 요소에서의 소요 시간

단위 : msec

| 배열종류 / 동작요소 | stuct 배열 | | integer배열 | |
|---|---|---|---|---|
| | 천 개 | 만 개 | 천 개 | 만 개 |
| 메시지 변환 | 3.1 | 4.3 | 3.2 | 4.2 |
| 다중전송 | 83.2 | 354.2 | 84.0 | 355.8 |
| 매개변수 변환 | 10.7 | 178.8 | 4.0 | 13.3 |
| 기타 | 13.5 | 251.9 | 5.8 | 17.8 |
| 합계 | 110.5 | 789.2 | 93.0 | 390.8 |

표 1의 전체 소요 시간은 클라이언트 응용 수준에서 측정한 수행 시간이며, 기타 부분은 전체 수행 시간에서 서버에서 측정한 시간들의 합을 제외한 시간이므로 TCP 전송 및 클라이언트 응용에서 소요되는 시간이 대부분이다. 따라서 다중 전송 및 메시지 변환에 소모되는 시간을 제외하면 기존 CORBA에서 소요되는 시간으로 인정될 수 있다. 물론 측정하기 쉽지 않은 고장인지를 위한 시간 그리고 각 쓰레드들 간의 동기화에 소요되는 시간도 고장감내를 위하여 피할 수 없는 성능 저하 요인이다. 측정 결과는 응용 수준에서의 성능 측정에서 예측할 수 있듯이 객체그룹을 지원하기 위하여 추가된 다중 전송 프로토콜이 전체 성능을 좌우함을 알 수 있게 한다.

## 6. 결 론

본 논문은 객체중개자 확장을 통해 객체그룹을 지원함으로써 고장감내를 제공하는CORBA 모델을 제안하고 구현하였다. 본 논문에서 제안하는 모델은 클라이언트와 서버 사이의 통신을 위하여 IIOP를 사용하며, 객체그룹 멤버들 사이에는 일치성 제어를 위해 전체 순서 지원 다중 전송 프로토콜을 이용하는 구조를 가진다. 이 같은 구조는 객체그룹 지원에 관한 투명성을 제공할 수 있기 때문에 기존의 CORBA와의 호환성을 제공할 수 있을 뿐 아니라 다중 전송 프로토콜에 독립적으로 구현될 수 있게 한다. 또 클라이언트 객체중개자를 다중 전송 프로토콜을 지원하도록 확장할 필요가 없기 때문에 소규모 자원을 제공하는 이동 컴퓨팅 응용에 쉽게 적응할 수 있다. 또한 본 논문은 객체중개자에서 객체그룹의 이름으로 객체그룹을 구성하기 때문에 응용 작성 시에 사용하여야 해야만 하는 인터페이스를 최소화할 수 있다. 이 같은 특성은 기존 CORBA에서 수행되는 서버 응용을 객체그룹을 지원하는 응용으로 쉽게 전환할 수 있게 한다. 또 본 논문에서 제안하고 있는 IDL의 readonly 구문은 성능 저하를 일으키는 가장 큰 요인이 되는 그룹 통신이 불필요하게 사용되는 것을 방지하기 때문에 응용의 특성에 따른 최적 성능을 제공할 수 있게 한다.

본 논문은 현재 능동복제 방식만을 지원하고 있지만 동작 모델이 수동복제 방식에서 사용하는 구조와 유사하기 때문에 수동복제 기능을 추가하는 것은 쉽게 구현될 수 있다. 수동복제의 지원 및 소규모 그룹에 최적 성능을 제공하는 다중전송 프로토콜을 개발하는 것 그리고 앞으로 제정될 OMG 표준을 만족하는 고장감내 CORBA를 개발하는 것은 향후 계획에 포함된다.

## 참 고 문 헌

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

[2] Ban, B., JavaGroups User's Guide, Department of Computer Science Cornell University, 1999.

[3] 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.

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

[5] Brose, G., JacORB Performance compared, http://www.inf.fu-berlin.de/~brose/jacorb/performance/results_09html, 1998.

[6] Budhiraja, N. et al, The Primary-Backup Approach, in Distributed Systems edited by Sape Mullender, 2nd Ed., Addition-Wesley, 1993.

[7] Defago, X., Schiper, A. and Sergent N., Semi-Passive Replication, IEEE Symposium on Reliable Distributed Systems, 1998.

[8] Ericsson, Iona Technologies and Nortel Networks, Fault Tolerant CORBA, OMG Document orbos/98-10-10, 1998.

[9] Eternal Systems and Sun Micro Systems, Fault Tolerance for CORBA Version 1.0 Initial Submission, OMG Document orbos/98-10-08, 1998.

[10] Felber, P., Garbinato, B., and Guerraoui, R., A CORBA Object Group Service, Technical Report 97-223, Ecole Polytechque Fdrale de Lausanne, 1997.
[11] 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.
[12] Highlander Communications et al., Fault Tolerant CORBA using Entity Redundancy, OMG TC Document orobos/98-10-09, 1998.
[13] Inprise Co., Programmer's Guide: Visibroker for Java Version 3.3, http://www.inprise.com/techpubs/books/vbj/vbj33/pdf_index.html, 1997.
[14] ISIS Distributed Systems and IONA Technologies, Orbix+ISIS Administrator's Guide, Isis Distributed Systems, 1995.
[15] IONA Technologies and ISIS Distributed Systems, An Introduction to Orbix+ISIS, IONA Technologies, 1994.
[16] 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.
[17] Little, M., Shrivastava, S., Understanding the Role of Atomic Transactions and Group Communications in Implementing Persistent Replicated Object, 8th IWPOS, 1998.
[18] Maffeis, S., Adding Group Communication and Fault-Tolerance to CORBA, USENIX, 1995.
[19] Miller, C. K., Multicast Networking and Applications, Addison Wesley, 1999.
[20] 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.
[21] 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.
[22] Objective Interface Systems, Fault Tolerant CORBA Through Entity Redundancy, OMG TC Document orbos/98-10-03, 1998.
[23] OMG, The Common Object Request Broker: Architecture and Specification, Revision 2.3, OMG, 1998
[24] Oracle Corporation, Fault Tolerance RFP, OMG TC Document orbos/98-10-13, 1998.
[25] Schneider, F. B., The State Machine Approach, Lecture Notes in Computer Science, 1987.
[26] Simon, B. and Spector, A., Fault-Tolerant Distributed Computing, Lecture Notes in Computer Science 448, Springer-Verlag, 1990.
[27] Stevens, W. R., TCP/IP Illustrated Vol. 1: The Protocols, Addison-Wesly, 1994.
[28] SUN Micro System, Java Native Interface Specification, SUN Document, 1997.
[29] 신범주, 이동현, 대등관계 복제객체 모델을 지원하는 분산 객체 프로그래밍 언어의 설계 및 구현, 정보과학회 논문지 제5권 제4호, 정보과학회, 1999.

신 범 주
1983년 경북대학교 전자공학과(학사). 1991년 경북대학교 컴퓨터공학과(석사). 1998년 경북대학교 컴퓨터공학과(박사). 1978년 ~ 현재 한국전자통신연구원(책임연구원). 관심분야는 분산객체시스템, 이동컴퓨팅, 네트워크 자료저장시스템 등

손 덕 주
1976년 서울대학교 수학교육과(학사). 1978년 한국과학기술원 전산학과(석사). 1978년 ~ 현재 한국전자통신연구원(책임연구원, 인터넷서비스연구부장). 관심분야는 분산 객체 시스템, 이동컴퓨팅, 이동단말 소프트웨어, 데이타베이스시스템, 네트워크기 반 자료저장시스템 등

김 명 준
1978년 서울대학교 계산통계학과(학사). 1980년 한국과학기술원 전산학과(석사). 1986년 프랑스 Nancy 1대학 전산학과(박사). 1986년 ~ 현재 한국전자통신연구원(책임연구원, 컴퓨터 소프트웨어 연구소장). 관심분야는 데이타베이스, 분산 시스템, 실시간DB 및 소프트웨어공학 등