2000년 한국정보처리학회 추계 학술논문발표집 제7권 제2호

고장 감내 자바RMI 객체 설계

이민석,* 윤태진,** 안광선*
*경북대학교 컴퓨터공학과
**경운대학교 소프트웨어공학과

e-mail:mslee@knight.knu.ac.kr

A Design of Fault Tolerance JavaRMI Object

MinSeok Lee,* TaeJin Yun,** KwangSeon Ahn
*Dept. of Computer Engineering, KyungPook National University
**Dept. of Software Engineering, KyungWoon University

요약

CORBA, DCOM, JavaRMI등과 같은 분산 객체 기술이 분산 응용의 신뢰성을 직접적으로 향상시키지는 못한다. 이러한 분산 객체 기술에 고장 감내성을 추가하기 위해서는 객체 단위의 복제 그룹 관리와 고장 탐지 및 회복 메커니즘이 필요하다. 본 논문에서는 고장 감내형 JavaRMI객체를 개발하기 위하여 고장 탐지와 그룹 관리를 위한 그룹관리자와 원격 인터페이스를 설계하고, 고장 감내성 클래스를 정의한다. 또한 고장 감내 객체의 투명한 그룹 참여를 위하여 Naming클래스와 RMIRegistry를 확장한다. 응용개발자는 고장 감내성 클래스를 상속함으로써 외부의 도움 없이 간단히 고장 감내 응용 객체를 개발 할 수 있다.

1. 서 론

최근 컴퓨터와 통신 기술의 발전에 따라, 분산 응용이 보편화되고 이를 개발하기 위한 많은 분산 객체 기술이 연구되었다. 대표적 표준 기술로는 CORBA[1], DCOM[2], JavaRMI[3] 등이 있지만 이러한 기술들이 응용의 신뢰성을 직접적으로 향상시키지는 못한다. 결과적으로 응용개발자들은 응용의 신뢰성과 가용성을 향상시키기 위하여 자체 메커니즘을 구현하여야만 한다. 최근 고장 감내 응용의 개발을 편리하게 하기 위한 재사용 가능한 도구들이 연구되어 왔다. 그러나 이들 중 많은 시스템이 통신 기법에 대해 프로그래머가 많은 것을 다루어야 하고, 시스템이 플랫폼에 종속적이며 또한 재사용성이 좋지 않아 매번 비슷한 고장 감내 코드를 만들어야 하는 등의 단점을 가진다.

본 논문에서는 JavaRMI시스템에서 신뢰성과 높은 가용성을 보장하기 위해 고장 감내 RMI객체와 원격 인터페이스를 설계하였다. 복제의 기본단위로 객체를 지원하고, 복제본은 같은 원격 인터페이스를 구현하며 같은 소스코드를 가진다. 객체 복제본들의 집합은 객체 그룹을 형성하며 그룹통신을 통하여 객체간의 일관된 상태유지와 고장탐지를 수행한다. 객체 복제본 간의 그룹 통신을 지원하기 위하여 그룹관리자 객체를 생성하여 고장 감내 객체에 포함함으로써 외부 서비스의 도움 없이 객체그룹을 형성할 수 있다. 본 시스템에서 응용 개발자는 상태 전달 인터페이스만을 구현함으로써 간단히 고장 감내 응용을 개발 할 수 있다.

2. 관련 연구

순수 자바 기반의 분산 객체 시스템인 Java RMI는 같은 호스트나 다른 호스트의 JVM(Java Virtual Machine)상에서 동작하는 원격객체에 의해 구현된 원격 인터페이스의 메소드를 로컬 객체의 메소드 호출과 같은 방식으로 호출 할 수 있게 한다. 이와 유사한 기술로는 RPC(Remote Procedure Call)가 있다.

분산 환경 하에서 고장 감내성을 향상시키기 위한 다양한 연구가 진행되어 왔다. 특히 분산 시스템의 표준이라 할 수 있는 CORBA(Common Object Request Broker Architecture)를 기반으로 한 최근 연구는 중복객체의 그룹관리 방식 및 메시지 전달방법 등에 따라 통합 방식, 인터셉트 방식, 서비스 방식으로 분류할 수 있다. 통합 방식은 CORBA의

ORB(Object Request Broker)아래에 신뢰성 있는 멀티캐스트 통신 서브시스템을 추가하여 새로운 ORB를 만드는 방식이다.[4] 이 방식에서 하위 레벨의 그룹 통신 계층을 이용하여 클라이언트의 요청을 복제본에 신뢰성 있게 멀티캐스트 함으로써 고장 감내 객체시스템을 구현하였다. 인터셉트 방식은 TCP/IP와 같은 하위 레벨의 통신 시스템에서 클라이언트 객체에 의해 만들어지는 시스템 콜을 가로채서 신뢰성 있는 멀티캐스트 서브시스템으로 전달하는 것이다.[5] 장점으로는 ORB의 수정 없이 응용에 투명하게 고장 감내성을 얻을 수 있다는 것이다. 서비스 방식은 그룹 통신을 ORB자체에 통합하지 않고 ORB상위의 CORBA 서비스로서 제공하는 것이다. 서버 측에서는 이 서비스를 이용하여 명시적으로 그룹을 만들고, 참여하여야 하며 클라이언트 측면에서는 서비스에 요청을 보내야 한다[6]. 서비스는 그룹의 관리와 클라이언트의 요청이 모든 그룹 복제본에 신뢰성 있게 전달되는 것을 보장하고 응답을 되돌려준다.

복제 시스템은 고장 감내 시스템에서 매우 중요한 요소이다. 복제 방법은 복제본의 상태를 어떻게 동기화 하는 가와 클라이언트 객체와 상호작용 방법에 따라 크게 "Cold Standby", "Worm Standby", "Hot Standby"로 나눌 수 있다[7]. Cold Standby 방식은 실행 시 프라이머리 객체와 백업객체 사이에 어떠한 통신도 발생하지 않는다. 이 방식은 통신비용이 매우 적은 반면 백업객체가 호출되었을 때 현재 프라이머리 객체의 상태를 가져와야 하므로 회복시간이 많이 소요된다. Warm Standby 방식은 프라이머리 객체가 호출된 요청과 내부상태를 주기적으로 백업객체에 전달한다. 실행시간 동안, 프라이머리 객체는 자신의 상태가 변할 때마다 백업객체의 상태를 갱신한다. 이 방식의 단점은 객체관리자에서 단일점 실패가 나타날 수 있고 고장 복구와 새로운 객체 생성 프로토콜이 복잡하다는 것이다. 통신비용은 Cold Standby에 비해 많으나 고장 복구시간은 적다. Hot Standby 방식은 다수개의 복재본이 모든 클라이언트의 요청에 대해 동시에 응답한다. 클라이언트는 모든 복제된 서버객체에 요청을 전달하고, 모든 서버 객체는 이 요청을 실행하고 결과를 클라이언트에 되돌려준다. 클라이언트는 첫 번째 리턴 값을 선택하거나, 모든 리턴 값을 받아서 그 중 하나를 선택할 있다. 만약 복제본 중 하나에 고장이 발생하더라도 다른 객체가 요청을 처리할 수 있으므로 복구과정이 필요 없다. 이전의 방식에 비해 통신비용이 가장 많이 소요되지만 복구시간이 가장 짧다.

3. 고장 감내 RMI 객체의 설계

본 논문에서는 고장 감내 RMI객체를 만들기 위하여 복제 방법으로 Warm Standby방식을 채택하였다. 이 방식에서는 모든 클라이언트의 요청은 프라이머리 객체로 전달되고, 프라이머리 객체는 이 요청을 처리한 후 변경된 상태를 백업객체에 멀티캐스트 한다. 또한 그룹관리와 상태전달, 고장 탐지를 위하여 원격 인터페이스를 정의하고 원격객체에 이 인터페이스를 구현한 그룹관리자를 내장함으로써 외부의 도움 없이 원격 객체간에 그룹통신을 가능하게 하였다.

3.1 구 조

분산 객체 시스템에 고장 감내성을 부여하기 위하여 그룹 관리, 고장 탐지, 고장 복구, 객체의 동일한 상태 유지 등의 기능이 필요하다. 본 시스템에서 프라이머리 객체는 다수개의 백업객체와 함께 그룹을 형성하며 그룹의 중재자 역할을 담당한다. 그룹의 모든 복제본은 그룹 관리와 고장 탐지를 담당하는 그룹관리자를 포함하며, 이들은 가상 링을 형성하여 객체 고장을 탐지한다. 프라이머리 객체의 고장 시, 백업객체 중 하나가 프라이머리 객체의 역할을 담당한다. 클라이언트는 원격 객체 호출을 위하여 RMIRegistry로부터 프라이머리 객체의 참조를 받아오기 때문에 모든 호출은 프라이머리 객체로 전달된다. 프라이머리 객체는 중요한 내부 상태가 변경되었을 때 그룹관리자를 통하여 백업객체로 상태를 멀티캐스트 함으로써 복제본의 상태를 동일하게 유지한다. 백업객체는 자신의 그룹관리자 객체를 통하여 프라이머리 객체에 등록함으로써 그룹에 참여한다. 프라이머리 객체는 백업객체에 그룹 뷰와 객체상태를 콜백 호출을 통하여 전달한다.

고장 감내 RMI시스템과 전체 구조는 그림 1과 같다.

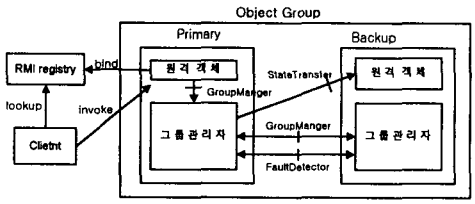

그림 1. 고장 감내 RMI시스템의 구조

각 복제본은 개발자가 구현한 원격 객체와 그룹관리자로 구성되어있으며, 수직선을 가진 화살표는 목표객체가 화살표에 표시된 원격 인터페이스를 지원한다는 것을 보여주기 위해 사용되었다. 원격 인터페이스 타입의 객체참조를 가진 클라이언트는 인터페이스에 의해 정의된 메소드를 호출할 수 있다.

본 시스템에서 객체의 투명한 그룹생성과 참여를 위하여 RMIRegistry를 수정하고, Naming클래스를 확장한 FTNaming 클래스를 정의한다.

3.2 고장 감내 RMI객체 시스템의 구성

본 시스템은 객체에 고장 감내성을 제공하는 고장 감내 클래스와 그룹관리를 위한 그룹 관리자 클래스로 구성되어 있으며 이러한 객체에 접근하기 위한

원격인터페이스를 제공한다.

● FTUnicastRemoteObject 클래스

표준 JavaRMI의 UnicastRemoteObject를 상속한 추상클래스로서 그룹관리를 위하여 그룹관리자 클래스를 포함하고 있다. 객체의 상태 전달을 위해 StateTransfer인터페이스를 구현한다. 고장 감내 객체를 구현하기 위해서는 이 클래스를 상속하여야 한다.

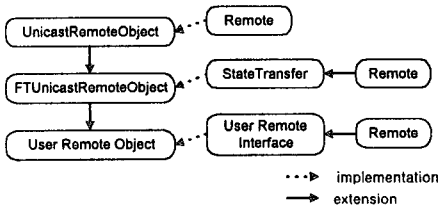

그림 2. 고장 감내 클래스의 상속도

● 그룹관리자

그룹관리자는 원격 인터페이스인 GroupManager와 FaultDetector 를 구현함으로써 그룹멤버간의 상호 호출을 통하여 그룹을 관리할 수 있다. 또한 FaultDetector 인터페이스를 통하여 객체 고장탐지를 수행한다.

```
public interface GroupManager extends Remote {
    public int join_Group(ServiceRef sr)
                    throws RemoteException;
    public void leave_Group(int crash_seq)
                    throws RemoteException;
    public void reset_GroupView(ServiceRefs gms, int id)
                    throws RemoteException;
}

public interface FaultDetector extends Remote {
        public boolean is_Alive()
                    throws RemoteException;
}
```

그림 3. FaultDetector와 GroupManager인터페이스

● StateTransfer 인터페이스

상태전달을 위하여 FTUnicastRemotObject에 의해 상속되며 응용개발자에 의해 set_State()와 get_State메소드가 구현되어야 한다. 그룹관리자는 이 메소드를 콜백 호출함으로써 객체에 상태를 전달한다.

```
public interface StateTransfer extends Remote {
    public Object get_State() throws RemoteException;
    public String set_State(Object obj)
                    throws RemoteException;
    public Remote get_GroupManager()
                    throws RemoteException;
}
```

그림 4. StateTransfer 인터페이스

4) RMIRegistry와 FTNaming클래스

표준 rmiregistry는 원격개체의 등록을 제한함으로 본 시스템은 이를 확장하여 원격객체가 등록할 수 있도록 수정한다. FTNaming클래스는 고장 감내 객체의 rmiregistry 등록과 투명한 그룹형성을 도와주는 유틸리티 클래스이다.

3.3 객체 등록

본 시스템에서는 rmiregistry에 등록된 이름 문자열을 기준으로 그룹을 형성한다. 가장 먼저 생성된 고장 감내형 객체는 FTNaming클래스의 bind메소드를 통하여 rmiregistry에 등록되고, 자신을 프라이머리 객체로 설정된다. 이후에 생성된 백업객체는 bind메소드에 의해 같은 이름을 가진 객체가 레지스트리에 존재하므로, 백업객체로 설정되고 프라이머리 객체의 그룹관리자에 그룹참여를 알린다. 이러한 과정은 그림 5와 같다.

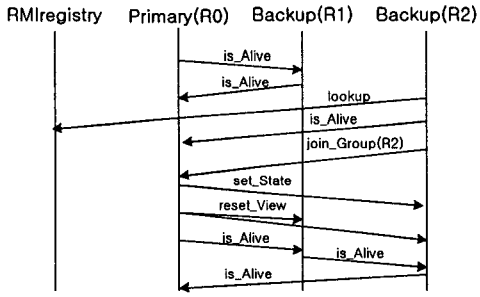

그림 5. 객체의 그룹 등록

새로 생성된 백업객체 R2는 주어진 이름 문자열을 가지고 FTNaming클래스의 bind 메소드를 호출한다. 이 메소드는 주어진 이름 문자열과 연관된 객체참조가 있는 지를 확인하여, 만약 없다면 프라이머리 객체로 R2를 설정하고 rmiregistry에 등록한다. 그러나 같은 이름을 가진 객체가 존재하면 프라이머리 객체의 GroupManager인터페이스 객체를 받아와서 그룹에 참여한다. 프라이머리 객체는 GroupManager인터페이스의 join_Group메소드가 호출되면, 백업객체의 GroupManager인터페이스의 set_State메소드를 호출하여 객체의 상태를 전달하고, 그룹의 모든 멤버에 새로 형성된 그룹 뷰를 전달하기 위해 reset_View()메소드를 호출한다.

3.4 고장 탐지 / 복구

객체 그룹의 멤버들은 그룹뷰의 순서에 의해 가상 링을 형성하며, 이 순서에 의해 일정시간 간격마다 자신의 상대편에게 FaultDetector인터페이스의 is_Alive함수를 호출함으로써 객체의 고장을 탐지한다. 만약 복제된 객체그룹의 멤버 중에서 고장이 탐지되면, 그룹의 중재자 역할을 하는 프라이머리객체에 GroupManager인터페이스의 leave_Group메소드 호출을 통하여 보고된다. 프라이머리 객체는 새로운

그룹뷰를 형성하기 위하여 그룹의 각 멤버의 FaultDector인터페이스의 is_Alive 메소드를 호출한다. 그림 6 은 백업객체의 고장탐지 시 새로운 복구 과정을 보여준다.

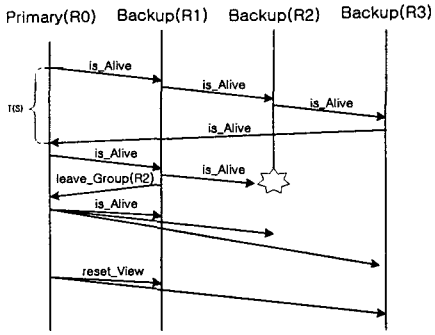

그림 6. 백업객체의 고장 복구

프라이머리 객체는 객체 그룹의 중재자로써 고장 탐지 시 고장 복구와 클라이언트의 모든 요청을 처리하는 중요한 역할을 담당한다. 그러므로 프라이머리 객체의 고장이 탐지되었을 때 백업객체 중 하나가 새로운 프라이머리 객체로 선출되어야 한다. 새로운 프라이머리객체 후보는 가상 링을 구성하는 순서에 따라 결정된다. 새로 결정된 프라이머리객체는 고장이 탐지된 이전의 객체참조를 대신해 자신의 객체참조를 RMIRegistry에 갱신한다.

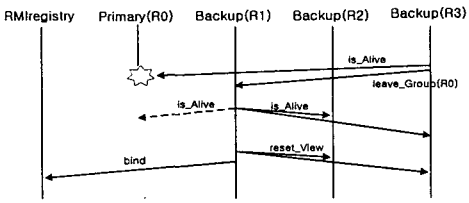

그림 7. 프라이머리 객체의 고장 복구

그림 7은 프라이머리 객체의 고장 시 새로운 프라이머리 객체를 어떻게 결정하는지 보여준다. 백업객체 R3는 프라이머리 객체 R0의 고장을 탐지하고, 백업객체 R1의 GroupManager 인터페이스의 leave_Group(R0)를 호출한다. 백업객체 R1은 프라이머리 개체 R0의 고장을 확인하기 위하여 is_Alive() 메소드를 호출 한 후 응답이 없으면, 자신을 프라이머리 객체로 설정하고 RMIRegistry에 자신의 객체참조를 등록한다. 새로운 프라이머리 객체 R1은 나머지 복제본에 is_Alive()메소드를 호출하여 새로운 그룹 뷰를 형성하고, reset_view()메소드를 호출하여 각 복제본에 전달한다.

4. 결 론

본 논문에서는 자바 RMI객체에 신뢰성과 가용성을 향상시키기 위하여 고장 감내 메커니즘을 구현한 그룹관리자와 원격 인터페이스를 설계하였다. 그룹관리자는 각 객체와 함께 생성되며 그룹의 고장탐지 및 복구와 상태 전달을 담당한다. 복제의 기본단위는 객체이며 이러한 복제본들은 그룹을 형성한다. 프라이머리 객체는 그룹의 고장 복구 시 중재자 역할을 담당하며, 프라이머리 객체 고장 시 그룹뷰에 의해 형성되는 가상 링의 순서에 의해 백업객체가 프라이머리 객체 역할을 대신한다. 프라이머리 객체는 RMIRegistry에 등록되며, 클라이언트와 백업객체는 레지스트리로부터 프라이머리 객체의 객체참조를 가져와서 원격 호출과 그룹 등록을 한다. 백업객체는 FTNaming 클래스를 이용하여 투명하게 그룹참여를 할 수 있다. 분산 응용 개발자는 고장 감내형 RMI클래스를 상속하여 상태 전달 인터페이스만을 구현함으로써 간단히 고장 감내 RMI객체를 개발할 수 있다. 향후 연구과제로서는 본 시스템의 구현과 성능평가 후, 다른 방식의 복제기법을 적용한 시스템과의 비교 평가가 필요하다.

참고문헌

[1] Object Management Group, "The Common Request Broker: Architecture and Specification," http://www.omg.org, 1999

[2] N. Brown, C. Kindel. Distributed Component Object Model Protocol - DCOM/1.0. *Internet Draft*, 1996

[3] Sun Microsystems. Remote Method Invocation Specification, 1999

[4] IONA and Isis, An Introduction to Orbix+Isis, IONA Technologies Ltd. and Isis Distributed Systems, 1994

[5] Narasimhan, P. Moser, L.E., Melliar-Smith, P.M., The interception approach to reliable distributed CORBA object, in Proceedings of USENIX Third Conference of Object-Oriented Technologies and System(COOTS'97), 1997

[6] P. Felber, B. Garbinato, R. Guerraoui, " owards Reliable CORBA: Integration vs. Service Approach," The Proceedings of 11th European Conference on Object-Oriented Programming, 1996

[7] Guang-Way Sheu, Yue-Shan Chang, Deron Liang, Shyan-Ming Yuan, and Winston Lo, "A fault-tolerant object service on CORBA," in Proceedings of the 17th International Conference on Distributed Computing, 1997

[8] N. Narasimhan, L. E. Moser, and P. M. Melliar-Smith, "Interception in the Aroma System," Proceedings of the ACM 2000 Java Grande Conference, 2000