객체 지향 실시간 운영체계상에서의 고장감내 객체 서비스

이은향*,김형환*,임동선*,정연호**,김영만**
ehlee@etri.re.kr* , nedoli@cs00.kookmin.ac.kr**, ymkim@kmu.kookmin.ac.kr**
한국전자통신연구원(ETRI)*
국민대학교 전산과학과**

Fault Tolerant Object Service on the Object-Oriented Real Time OS

Eun-Hyang Lee*, Hyung-Hwan Kim*, Dong-sun Lim*,Young Man Kim**, Yun-Ho Chung**
ETRI*
Department of Computer Science, Kookmin University**

요약

본 논문에서는 CORBA 의 고장 감내(fault tolerance) 방식을 이용하여 교환기 시스템을 위한 객체 지향 실시간 운영체계상에서의 고장 감내 객체서비스 구현을 목표로 한다. 이를 위해 먼저 CORBA 의 고장 감내 방식을 분석하고,전자 교환기에 적합한 실시간 고장 감내 방식에 대하여 제안한다. 이 방식에서는 교환기내의 이중화 시스템을 이용하여 객체 단위에서 고장 감내 기능을 제공하게 된다.

1. 서론

분산 환경에서 운용되는 CORBA(Common Object Request Broker Architecture)[OMG 95] 기반의 응용 프로그램은 ORB 를 통해 협동하는 클라이언트 프로그램과 서버 객체로 구성된다. 클라이언트는 분산 시스템을 통해 보다 신뢰성이 있고 최적화된 서비스를 받기를 원하는데, 기존의 CORBA 환경에서는 점 대 점(point to point)방식이기 때문에 고장 감내(fault tolerance)를 필요로 하는 분산 어플리케이션에는 적당하지 않다. 이러한 제약 때문에 새로운 분산 처리 플랫폼에 관한 연구가 활발히 진행되고 있다.

한편, 기존의 전자 교환기에서는 장애가 발생할 경우 신속하게 고장을 발견하고 시스템 기능을 회복하여 새로 발생하는 서비스 요청이 거부되거나, 진행중인 서비스가 중단되는 일이 없도록 하는 고장 감내 하드웨어가 설계 및 구현되어 있다.

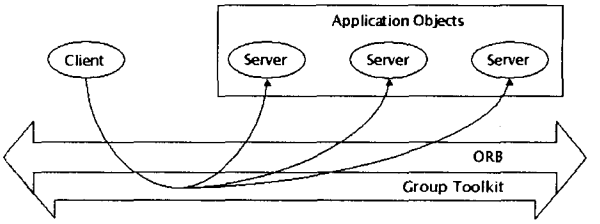
본 논문에서는 CORBA 에서의 고장 감내형 서비스를 제공하는 데 사용되는 기존 방식에 대하여 설명하고 이 중에서 통합(Intergration)방식을 채용하여 교환기구조에 적합하도록 개조한 복제 이중화 서비스 방식을 소개하고자 한다. 먼저 2 절에서는 CORBA 고장 감내 서비스 방식들에 대하여 살펴본다. 3 절에서는 교환기용 복제 이중화 서비스에 대하여 기술한다. 4 절에서는 결론 및 향후 연구 과제에 대해 기술한다.

2. CORBA 고장 감내 서비스

2.1 통합 방식(Integration Approach)

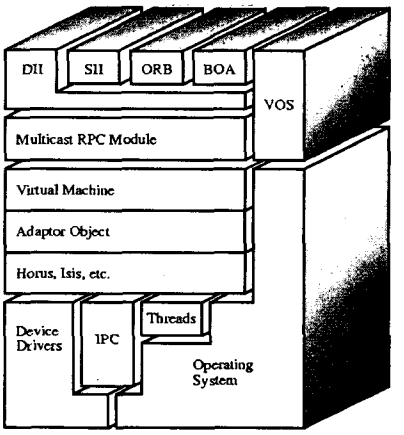
[그림 1]에서 보듯이 통합 방식은 기존의 OMG ORB 에 그룹 툴킷(group toolkit)을 통합시킨 새로운 형태의 ORB 가 복제객체 그룹관리 및 멀티캐스팅과 같이 장애 감내성 제공에 필요한 기본 기능들을 제공하는 방식이다 [IONA 98,ORACLE 98].

**본 논문의 연구는 1999 년도 한국전자통신연구원(ETRI)의 위탁연구과제로 수행되었음.



[그림 1] 통합 방식

대표적인 연구사례로 Orbix+Isis[Maffeis 99]와 Electra [Maffeis 99]가 있다. [그림 2]는 Electra 구조를 나타내고 있다.



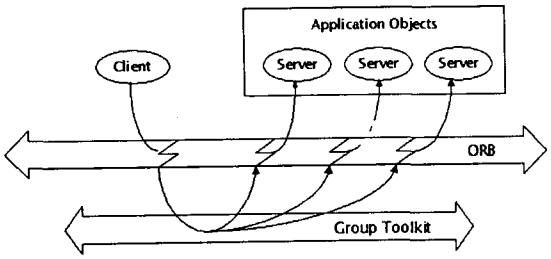
[그림 2] Electra 의 구조

Orbix+Isis 는 Orbix™ CORBA-compliant C++ 개발 환경과 Isis Reliable runtime 기술을 통합한 것이다. Orbix 는 표준 객체지향 프로그래밍 환경, 인터페이스 정의의 추상화

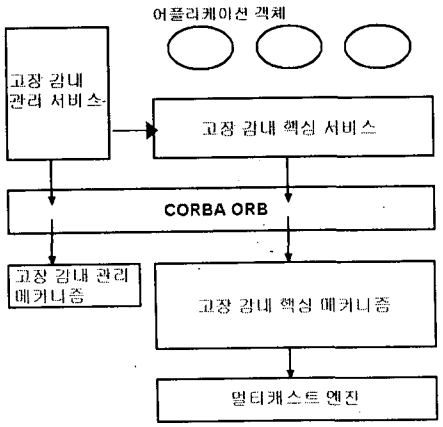
등의 분산 객체지향 CORBA 환경의 이점을, Isis 는 프로세스 그룹, 뷰 관리, 상태 전송, request ordering, virtual synchrony 를 제공한다. Orbix+Isis 의 특징은 클라이언트에 투명하여, 클라이언트 코드의 수정 없이 고장 감내 기능을 제공하는 그룹 객체들과 함께 수행할 수 있다.

2.2 방수 방식(Interception Approach)

방수 방식은 SUN 사에서 제안한 방식[SUN 98]으로, 기존 ORB 를 수정하지 않고 클라이언트가 보낸 메시지를 그룹 툴킷이 가로채서 고장감내 기능을 구현한다. [그림 3]은 방수방식의 기본 구조 및 기능을 보여준다. SUN 사의 고장감내 모델 하부 구조는 객체 복제와 고장 감내 서비스를 제공하기 위해, [그림 4]에서 보듯이 고장 감내 관리 서비스(fault tolerance management service), 고장 감내 핵심 서비스(fault tolerance core service), 고장 감내 핵심 메커니즘(fault tolerance core mechanism)의 3 단계 제어 방식을 채용하고 있다.



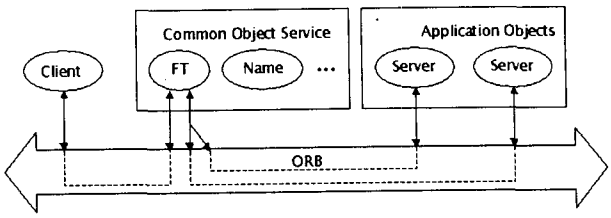
[그림 3] 방수 방식



[그림 4] 장애 감내 하부 구조

2.3 서비스 방식(Service Approach)

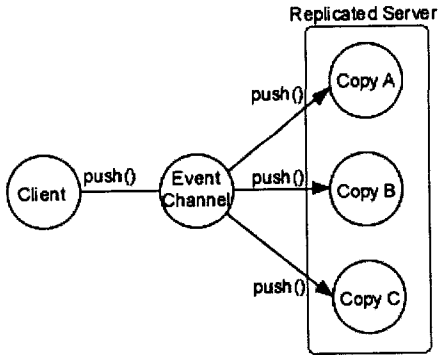
서비스 방식[Felber 97]에서는 기존의 ORB 를 수정하는 대신 객체 그룹을 관리하고 장애 감내 서비스를 위해 필요한 기능을 제공하는 COS(Common Object Service) 객체를 정의하여 사용하고 있다. 예로는 OGS(Object Group Service)[Felber 96]와 DOORS(Distributed Object Oriented Reliable System)[Chung 98]이 있다. [그림 5]는 서비스방식의 구조 및 객체간 운용 관계를 보여준다.



[그림 5] 서비스 방식

2.4 이벤트 서비스 방식(Event Service Approach)

이벤트 서비스방식[Felber 97]에서는 공급자는 이벤트 채널을 이용해서 이벤트 데이터를 생성하고 소비자는 이벤트 채널을 통해 전달된 데이터를 처리한다. 이벤트 채널은 다수 공급자와 다수 소비자 사이에 비동기 통신을 제공하는 표준 CORBA 객체이다. 이벤트 서비스는 PUSH 모드와 PULL 모드라는 두 가지 형태의 통신모드를 제공한다. [그림 6]에서와 같이 PUSH 모드에서는 공급자가 소비자에게 이벤트 데이터를 제공하고 초기화를 담당하며, PULL 모드에서는 소비자가 공급자에게 이벤트 데이터를 요청한다.



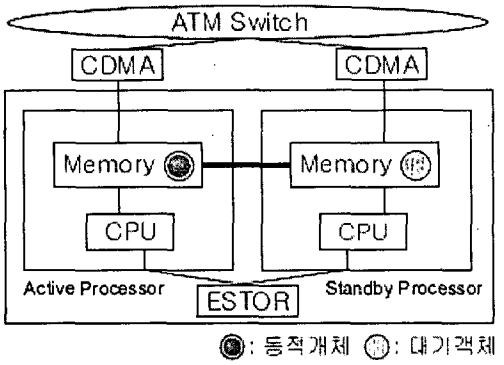
[그림 6] 중복된 서비스 객체를 위한 이벤트 서비스

3. 이중화 서버

이 절에서는 warm(active/standby)방식을 채택하여 전자교환제어 시스템에 적합한 고장감내 객체서비스에 대하여 기술하고자 한다. [그림 7]은 전자교환기내의 하드웨어 이중화 구조를 나타내고 있는 데 교환기에서 운용되는 고장 감내기능은 시스템 레벨에서 발생하는 고장에 대한 신속한 검출 기능과 검출된 고장에 대하여 자동으로 복구하는 기능과 정의되지 않는 장애에 대비하여 신속한 재구성 기능을 포함한다.

이중화된 전자 교환기는 기본적으로 [그림 7]과 같은 구조를 가지며, 관리 모듈을 두어서 위와 같은 이중화 시스템의 장애를 대비해서 warm 이나 hot 방식을 이용해 설계한다.

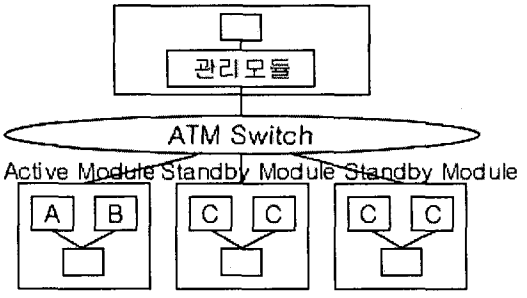
Active 상태에 있는 프로세서의 객체(동적객체)는 요청을 처리하고 처리된 결과는 요청자에게 돌려주며 처리 후 상태 정보는 ESTOR 에 저장한다. 이는 동적객체가 매 요청 처리 후 대기상태에 있는 프로세서상의 객체(대기객체)에게 상태정보를 전달하는 것을 방지함으로써 부하를 줄이고 또한, 고장감내를 구현하는 역할을 한다.



[그림 7] 전자 교환기 내 이중화 구조

동적객체는 매 요청 후 대기객체에게 상태를 넘기는 것이 아니라, 10-20 회 정도 처리 후 상태 정보를 대기객체에게 전달한다. 주로 하드웨어적인 영구적인 결함에 대해서는 active processor 가 스스로 이러한 결함을 검출하여 신속히 standby processor 에게 제어를 이전하여 시스템의 동작의 연속성을 유지한다.

[그림 7]과 같은 이중화 시스템에서 동작객체와 대기객체가 동시에 고장날 경우에 대비해서 복제 이중화 시스템을 [그림 8]과 같이 구성한다.



[그림 8] 복제 이중화 전자 교환기 구성도

그림에서 이중화 전자 교환기네트워크는 세개의 교환제어 모듈로 이루어져 있으며 각각의 모듈내에는 한쌍의 프로세서가 놓여있다. 따라서 각 프로세서마다 하나의 복제 객체가 배치된 세쌍의 복제객체는 동일 그룹내의 객체로서 세 교환기 모듈내에 배치된다. 이 중에서 왼쪽의 교환기내의 한쌍 객체가 active 객체로서 요구 메시지를 처리하며 다른 교환기내의 객체들은 standby 객체로서 대기 상태에 들어간다. 한편, 왼쪽의 교환기내에서 왼쪽 객체는 동작객체, 오른쪽 객체는 대기 객체로서 배치되어서 동작객체에 장애가 발생할 경우, 실시간 절체에 의하여 동작객체로 변환되어서 교환기 모듈 외부에서는 전혀 고장 발생 영향이 전파되지 않도록 한다.

Active 객체는 요청을 한 후에 새로운 상태를 자신의 ESTOR 에 저장을 한다. Active 모듈 전체가 동시에 고장 발생할 경우를 대비해서 ESTOR 에 있는 정보를 주기적으로 관리 모듈 ESTOR 에 저장한다. 저장된 정보는 주기적으로 standby 모듈에 전송한다. Active 모듈이 장애를 발생했을 경우 관리 모듈은 standby 모듈 중 하나를 선택하여 active 모듈로 절체한 후 서비스를 수행한다. 관리 모듈은 시스템 내에서 발생하는 장애를 수집하여 관리하며 이를 보고하는 기능이 포함되어 있다. 교환기 시스템은 고 가용성을 추구하는 실시간 시스템으로 최소한의 시스템 다운 시간을 가져야 하는 제약 사항이 있다. 이를 위해서 시스템의 장애 관리 기능이 중요한 역할을 한다. 즉, 항상 시스템의 상태를 감시하여 시스템의 보수를 요하는 장애가 발생할 때는 이를 즉시 운영자에게 알려 조치를 취하는 기능이 필요하다.

4. 결론

본 논문에서는 기존의 이중화 전자 교환기상의 객체지향 플랫폼에서의 고장감내 서비스 구조를 제안하는 것에 중점을 두었다. 본 논문에서 제안한 warm 방식과 hot 방식과의 비교분석을 하는것과 복제 이중화 시스템을 설계 및 구현하는 것이 향후 과제로 남아있다.

참고 문헌

[Felber 97] P. Felber, R. Guerraoui, and A. Schiper, "Replicating objects using the CORBA event services", Proceedings of the IEEE Computer Society Workshop on Future Trends in Distributed Computing Systems(FTDCS-6), pp.14-19, Tunis, Tunisia, October 1997.

[IONA 98] Ericsson, IONA, and Nortel, "Fault Tolerant CORBA", Fault Tolerance Joint Initial Submission, OMG document orbos/98-10-10, October 1998.

[Chung 98] P.E. Chung, et al., "DOORS : Providing Fault Tolerance for CORBA Applications", poster session of Middleware'98.

[SUN 98] Eternal and SUN, "Fault Tolerance for CORBA", Fault Tolerance Joint Initial Submission, OMG document orbos/98-10-08, October 1998.

[Oracle 98] Oracle, "Fault Tolerance RFP", OMG Document orbos/98-10-13, October 1998.

[Maffeis 99] S. Maffeis and S. Landis, "Building Reliable *Distributed System* with CORBA", to appear in Theory and Practice of Object Systems, John Wiley Publisher, NY.

[Felber 96] P. Felber, B. Garbinato and R. Guerraoui, "The Design of a CORBA Group Communication Service", Proceeding of the 15th IEEE symposium on Reliable Distributed Systems, p150-159, October 1996

[OMG 95] OMG, "The Common Object Request Broker : Architecture and Specification", July 1995.