CORBA에서의 그룹 통신 지원을 위한 OCI 기반 프레임웍

남덕윤° 이동만
한국정보통신대학원대학교 공학부
{paichu, dlee}@icu.ac.kr

OCI-based framework to support group communication in CORBA

Dukyun Nam° Dongman Lee
School of Engineering, Information and Communications University (ICU)

요 약

CORBA는 이종의 분산 컴퓨팅 환경에서 재사용성, 이식성, 상호 운용성을 유지하며 통합할 수 있는 환경을 제공한다. 그러나 표준 CORBA는 객체 복제를 이용하여 지원할 수 있는 결함 감내와 고 가용성을 지원하지 못한다. 지금까지의 CORBA 기반 그룹 통신에 관한 연구들은 CORBA 애플리케이션 프로그래머가 그룹 통신 프로토콜을 직접 이용할 수 있는 방법이 없었다. 또한 CORBA 또는 OS의 수정이 요구되거나, 기존의 다양한 그룹 통신 프로토콜을 적용할 수 없었다. 본 연구에서는 상호 운용성, 기존 그룹 통신 프로토콜의 재사용, ORB와 OS에 대한 독립성을 유지하고, 유연성 있는 하부 프로토콜 적용을 가능하게 하는 OCI를 확장함으로써, 표준 CORBA의 수정 없이 다양한 그룹 통신 프로토콜을 적용할 수 있는 그룹 통신 프레임웍을 제안한다.

1. 서론

CORBA (Common Cbject Request Broker Architecture) 는 이종의 분산 컴퓨팅 환경에서 객체 복제 (object replication) 를 이용하여 결함감내 (fault tolerance) 와 고 가용성 (high availability) 을 지원할 수 있다. 이러한 복제들의 모임인 객체 그룹 (object group) 은 일반적인 작업을 처리하는데 있어서, 그룹 내의 멤버들끼리 협동적으로 일을 처리한다 [13]. 한편 신뢰성과 결함 감내를 지원하는 분산 애플리케이션을 구현하기 위해서는 그룹 멤버들 사이의 일관성 (consistency) 이 유지되어야 한다 [10]. 여기서 멤버들 사이의 일관성은 그룹 통신 서비스 (Group Communication Service : GCS) 를 이용해 보장할 수 있다.

CORBA에서 GCS를 지원하기 위한 연구들은 최근까지 계속되고 있다. 일반적으로 이 연구들은 통합 방식 (integration approach), 서비스 방식 (service approach), 가로채기 방식 (interception approach) 으로 크게 3가지 방식으로 구분된다. 이 방식들의 예로는 각 방식마다 Electra [13], Object Group Service (OGS) [7, 8], Eternal [9] 을 들 수 있다. 이 방식들은 그룹 통신 프로토콜을 CORBA에 적용하는 데에 있어서 하부 프로토콜에 대한 인터페이스를 지원하지 않기 때문에, 애플리케이션 프로그래머가 프로토콜을 직접 활용할 수 없다.

CORBA는 이종의 분산 환경에서 분산된 객체들이 서로 통신을 할 수 있도록 이식성 (portability) 과 상호 운용성 (interoperability) 을 보장한다. 이런 측면에서 특정 그룹 통신 프로토콜만이 CORBA에 적용되는 것은 CORBA 그룹 애플리케이션의 요청을 만족 시키지 못한다. 이를 해결하기 위해서는 표준 CORBA 인터페이스를 통해 다양한 그룹 통신 프로토콜을 적용할 수 있는 일반적인 그룹 통신 프레임웍이 있어야 한다. 본 논문에서는 상호 운용성, 기존의 다양한 그룹 통신 프로토콜에 대한 재사용성, ORB와 OS에 대한 독립성을 유지하고, 유연하게 하부 프로토콜을 적용할 수 있는 OCI (Open Communication Interface) 를 확장함으로써, 기존의 CORBA를 수정하지 않고 그룹 통신을 적용할 수 있는 프레임웍을 제안한다.

2. 관련연구

이 장에서는 CORBA에서 그룹 통신을 지원하는 방법에 관한 연구들을 살펴보고자 한다. Electra [13] 는 CORBA의 BOA (Basic Object Adapter) 를 확장함으로써 그룹 통신 서비스를 지원하고자 했다. 이 시스템은 ORB와 그룹 통신 시스템 사이에 중계 객체 없이 바로 연결이 되어 있으므로, 구현과 성능 측면에서 효율적이라 할 수 있다. 그러나 그룹 레퍼런스 (group reference) 와 그룹 관리 (group management) 를 지원하기 위해 ORB를 수정해야 한다는 단점이 있다.

OGS [7, 8] 는 그룹 통신을 위한 새로운 CORBA 객체 서비스로 디자인 되었다. 이 방식은 ORB에 대해 독립적이며, CORBA 객체 서비스로서 이식성을 보장하지만, 기존의 그룹 통신 시스템을 이용할 수 없으며, 성능 측면에서 비효율적이라는 단점을 가지고 있다.

Eternal [9] 은 그룹 통신 시스템의 메시지에 해당하는 IIOP 메시지를 가로채서 복제 메니저에게 전송한다. 이 방식은 인터셉터 [10]를 이용하기 때문에 ORB를 수정할 필요가 없으나, 인터셉터가 시스템 콜 수준에서 구현되어 있기 때문에 OS에 종속된다는 단점이 있다.

CORBA에 TCP/IP 외에 멀티캐스트 프로토콜을 적용한 연구 [1] 에서는 OCI를 이용해서 IP 멀티캐스트를 CORBA에

적용했다. OCI는 CORBA에서 TCP/IP를 대체할 수 있는 인터페이스를 제공한다. 이 모듈은 클라이언트/서버 모델의 통신에서 연결 초기화 부분과 통신 부분을 나누어 놓은 것이다. CORBA는 전송 수준 (transport layer) 에 대해 연결 지향 (connection oriented), 신뢰성 있는 데이터 전송 (reliable data transport), 스트림으로 전송 가능한 데이터 (transported data as a stream), 연결 손실에 대한 통지 (notification of connection loss) 를 해야 한다고 정하고 있다 [14]. 이에 따라 IP 멀티캐스트를 CORBA에 적용할 때, 긍정 응답 (acknowledgement) 와 소실된 패킷에 대한 재전송 (retransmission) 기능을 추가하였다. 그리고 IOR (Interoperable Object Reference) 에 IP 멀티캐스트 주소와 시퀀스 번호를 추가하였다. 이 연구는 Java IP 멀티캐스트 API를 그룹 함수로써 직접 이용하기 때문에 동적 그룹 멤버쉽을 지원하지 않으며, CORBA에서 다양한 그룹 통신 프로토콜을 적용할 수 없다.

마지막으로 MGIOP (Multicast Group Internet Inter-ORB Protocol) 엔진에 대한 내용이 MGIOP 명세서 [5] 에 제안되었다. MGIOP는 그룹 ID와 도메인 ID를 포함하며, 특히 GIOP (General Inter-ORB Protocol) 메시지 자체를 포함하고 있다. MGIOP 엔진은 서로 다른 그룹 통신 프로토콜을 동시에 지원할 수 있다. 그러나 이 엔진은 ORB에 포함되거나 ORB와 연결이 되어야 하므로, ORB를 수정 해야 한다는 단점이 있다.

3. CORBA 내에서의 그룹 통신을 위한 제안

3.1. Open Communication Interface

OCI는 CORBA에서 하부 프로토콜을 교체할 수 있는 인터페이스를 제공한다. 인터페이스들은 Buffer, Acceptor, Transport, Connector, Connector Factory, Registries, Info 객체 등이다 [6]. Buffer는 데이터를 가지고 있으며, 클라이언트와 서버 사이의 통신에 사용하는 위치 카운터를 관리한다. Info 객체들은 Acceptor, Transport, Connector, Connector Factory에 대한 정보들을 가지고 있으며, 각 객체마다 자신의 Info 객체를 가지고 있다. 클라이언트와 서버가 활성화 될 때, 객체 어댑터 (Object Adapter; OA) 내의 Acceptor Registry는 Acceptor를 생성하고, ORB core 안에 존재하는 Connector Factory Registry가 Connector Factory를 생성한다. 서버 측에서는 Acceptor가 IOR에 들어갈 프로파일 정보를 만들고, OA는 IOR을 생성한다. 한편 클라이언트 측에서는 Connector Factory가 IOR의 내용에 따라 Connector를 생성한다. 이 후, Connector와 Acceptor는 각각의 Transport 객체를 초기화하여 통신을 맺는다.

3.2. 그룹 통신을 위한 OCI 확장

CORBA 객체에 GCS를 적용하기 위해서는 그룹 주소 (group address), 그룹 멤버쉽 (group membership), 순서화 (ordering) 의 3가지 측면이 고려되어야 한다. 본 연구에서는 하부 그룹 통신 프로토콜이 메시지에 대해 신뢰성 있는 전송을 보장한다고 가정한다.

그룹 주소는 클라이언트가 그룹을 하나의 객체와 같이 인식함으로써, 투명성을 보장하여 그룹 객체들이 작업을 수행할 수 있도록 해 준다. 여기서 제안하는 OCI 확장은 그룹 통신 프로토콜에서 제공하는 그룹 구별자를 사용하여 그룹 객체 레퍼런스를 만든다. 그룹 객체 레퍼런스는 IOR로써 표현되며 OCI의 transport에 그룹 구별자를 확장한다. 클라이언트는 IOR에 있는 정보를 참조하여 그룹으로 통신을 위한 연결을 할 수 있다. GIOP가 끝단 (end-point) 의 정보인 그룹 주소

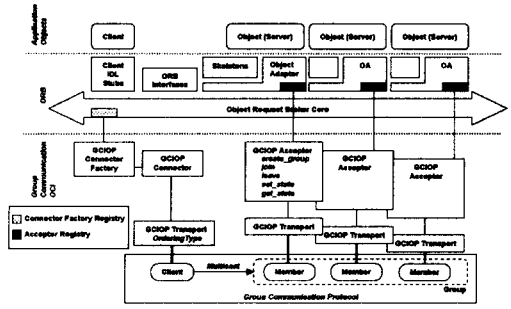

그림 1. 그룹 통신 프로토콜과 확장 OCI 구조

를 가지고 있지 않는 추상적인 프로토콜이기 때문에, 통신을 할 때에는 GIOP에 호환되는 프로토콜이 필요하다. 실제로 TCP/IP를 전송 프로토콜로 사용하는 경우에는 IIOP (Internet Inter-ORB Protocol) 를 구현해서 사용한다. 이에 이 연구에서는 그룹 통신을 지원하기 위해 GIOP 에 대한 구현 프로토콜인 GCIOP (Group Communication Inter-ORB Protocol) 을 정의했다.

그룹 멤버쉽에서는 그룹 내의 멤버들 사이의 일관성 유지를 위해 멤버 뷰를 관리한다. 이에 확장한 인터페이스에서는 그룹 초기화, 소멸, 가입, 탈퇴 등의 그룹 함수를 제공한다. 실제로 이 함수들은 OCI의 Acceptor에 추가된다. 상태 전송 (state transfer) 인터페이스는 Acceptor를 통해 제공되며, 상태의 정보는 CORBA 객체 타입으로 Acceptor Info 객체에 저장된다.

순서화는 하나 이상의 클라이언트들이 그룹과 통신할 때 멤버들 사이의 일관성을 유지하기 위해 필요한 요소이다. 클라이언트는 Transport Info에 있는 순서화 타입의 값을 변경함으로써 순서화 타입을 조정할 수 있다. 여기서 기본 순서화 값은 Total 순서화로 한다. 그림 1은 CORBA에서의 하부 그룹 통신 프로토콜과 OCI 구조를 보여준다.

3.2.1. IOR 확장

Tag	Group Name	Host Name	Object Key

그림 2. 그룹 통신을 위한 Tagged 프로파일

서버 측의 객체가 활성화 되었을 때, CORBA는 IP 주소와 포트 번호와 같은 전송 정보를 포함한 IOR을 만든다. 이와 같이 그룹 IOR은 그룹 이름과 각 멤버의 호스트 이름을 포함한다. IOR은 서버측에서 초기 실행시 만들어 지며, 클라이언트 애플리케이션 객체에 의해 분석된다.

그림 2는 GCIOP에서 사용되는 Tagged 프로파일 형식이다. Tag는 고정 값으로 표현되는 객체 간의 통신을 위해 사용되는 프로토콜을 나타낸다. '01'은 이미 IIOP에서 쓰이고 있으므로, 그룹 통신을 위해서는 이 외의 다른 값을 사용한다. 그룹 이름 (Group Name) 은 ORB는 물론, 하부 그룹 통신 프로토콜에서도 그룹 구분자로 사용된다. 호스트 이름 (Host Name) 은 멤버 자체의 호스트 이름을 나타내며, 객체 키 (Object Key) 는 특정 객체의 인스턴스 (instance) 를 나타낸다.

3.2.2. OCI 정보 구조와 인터페이스

그룹 통신 OCI 부분의 모든 구성 요소들은 각각 정보 객체를 가지고 있다. 모든 Info 클래스들은 기존 OCI의 Info 클래

스로부터 상속을 받아 확장된 것이다. 그림 3은 그룹 통신을

```
module OCI
{
module GCIOP
{
enum OrderingType {Total, Causal};

interface ConnectorInfo : OCI::ConnectorInfo{
   readonly attribute string group_name;
};

interface AcceptorInfo : OCI::AcceptorInfo{
   readonly attribute string group_name;
   readonly attribute string host_name;
   readonly attribute Object state;
};

interface ConnectorTransportInfo : OCI::TransportInfo
{
   readonly attribute string group_name;
   readonly attribute string host_name;
   attribute OrderingType ordering_type;
};

interface AcceptorTransportInfo : OCI::TransportInfo
{
   readonly attribute string group_name;
   readonly attribute string host_name;
};

}; // End module OCI::GCIOP
}; // End module CCI
```

그림 3. 정보 객체의 IDL 명세서

```
module OCI
{
module GCIOP
{

typedef IOP::IOR IOR;
enum OrderingType {Total, Causal};

interface Acceptor : OCI::Acceptor
{
   void create_group(in string group_name);
   void join(in string group_name);
   void leave(in string group_name);
   void set_state(in Object state);
   Object get_state();
};

}; // End module GCIOP
}; // End module OCI
```

그림 4. 그룹 통신을 위한 OCI의 IDL 명세서

위한 OCI Info 클래스의 IDL 명세서 (specification) 이다. ConnectorInfo는 그룹 이름을 저장하고 있으며, Connector는 메시지가 그룹에 보낼 때 ConnectorInfo를 사용한다. ConnectorInfo는 클라이언트 측의 OCI로 관리한다.

서버 측의 OCI는 그룹 이름과 호스트 이름을 가지고 있는 AcceptorInfo를 관리한다. 그룹 이름은 서버가 그룹을 만들거나 가입할 때 사용하는 것이며, 호스트 이름은 서버 자체의 호스트 이름을 나타낸다. 그룹 및 호스트 이름은 GCIOP 프로파일 정보를 만들 때 사용된다. 상태 (state) 는 그룹 멤버의 상태를 나타내며, 새로운 멤버가 그룹에 가입 했을 때 상태 전송 (state transfer) 을 위해 사용된다.

그림 4는 그룹 통신을 위한 함수가 추가 된 OCI의 IDL 명세서이다. 그룹 통신 OCI의 Acceptor 인터페이스는 애플리케이션 객체들만이 이용할 수 있다. 클라이언트 객체는 ConnectorFactory에 의해 Connector를 생성하고, 이 때 'set_type' 함수를 이용하여 순서화 값을 결정한다. Connector에는 추가되는 사항이 없다. Acceptor는 그룹 멤버쉽과 상태 전송을 위한 인터페이스를 제공한다.

4. 토론 및 결론

본 연구에서의 방법은 이식성 있는 인터셉터 (Portable Interceptors) [12] 의 한 부분인 네트워크 레벨 인터셉터 (network interceptor) 를 고려했었다. 왜냐하면 네트워크 레벨 인터셉터는 CORBA에서의 전송 프로토콜을 유연하게 적용할 수 있게 해주기 때문이다. 그러나 네트워크 레벨 인터셉터는 최근 명세서 [11] 에서 언급되지 않았다. 이에 네트워크 레벨 인터셉터와 유사한 기능을 지원해 주는 것으로 OCI와 MGIOP 엔진을 고려하게 되었다. OCI는 IN/CORBA 명세서 [3, 6] 에서 다양한 전송 프로토콜을 적용할 수 있도록 디자인 되었다. 결국 본 연구에서 제안한 프레임웍은 ORB와 OCI를 수정하지 않고 다양한 그룹 통신 프로토콜을 적용할 수 있으며, 보다 자세한 구현 및 동작 과정은 [2] 에서 볼 수 있다.

우리는 그룹 통신 모델로 일대다 (one to many) 통신을 가정했다. 송신자 (sender) 는 Connector 위에 위치하며, 다수의 수신자 (receiver) 는 Acceptor 위에 위치한다. 애플리케이션 객체들이 서버와 클라이언트 역할을 동시에 할 수 있는 모델 (reactive model) 이라면, 제안한 디자인은 다대다 (peer to peer) 통신에 적용할 수 있다. 결국 본 연구에서 제안한 프레임웍은 CORBA내 그룹 통신 지원은 물론, ORB 수정 및 OS에 대한 의존성 없이 기존의 그룹 통신 시스템을 지원한다.

5. 참고 문헌

[1] A. T. van Halteren, A. Noutash, L. J. M. Nieuwenhuis, M. Wegdam, "Extending CORBA with Specialised Protocols for QoS Provisioning", *Int'l Symp. on Distributed Objects and Applications*, Sep. 1999, pp. 318-327.

[2] D. Nam, D. Lee, H. Y. Youn, and C. Yu, "Group Communication Support for CORBA using OCI", *Twelth IASTED Int'l Conf. on Parallel and Distributed Computing and Systems*, Nov. 2000 (to appear).

[3] *IN/CORBA Initial v1.1*, telecom/98-06-03, June 1998.

[4] Kenneth P. Birman, "The Process Group Approach to Reliable Distributed Computing", *CACM, 36(12)*, Dec. 1993, pp. 37-53.

[5] L. E. Moser, P. M. Melliar-Smith, P. Narasimhan, R. R. Koch and K. Berket, "Multicast Group Communication for CORBA", *Int'l Symp. on Distributed Objects and Applications*, Sep. 1999, pp. 98-107.

[6] *ORBacus Web Site:* http://www.ooc.com/ob/, Object-Oriented Concepts, Inc.

[7] P. Felber, B. Garbinato, R. Guerraoui, "The Design of a CORBA Group Communication Service", *Proceedings of the 15th IEEE Symp. on Reliable Distributed Systems*, Oct. 1996, pp. 150-159.

[8] P. Felber and R. Guerraoui, "Programming with Object Groups in CORBA", *IEEE Concurrency, 8(1)*, Jan.-March 2000, pp. 48-58.

[9] P. Narasimhan, L. E. Moser, and P. M. Melliar-Smith, "Exploiting the Internet Inter-ORB Protocol Interface to Provide CORBA with Fault Tolerance," *Third USENIX Conf. on Object-Oriented Technologies and Systems*, June 1997, pp. 81-90.

[10] P. Narasimhan, L. E. Moser, and P.M. Melliar-Smith, "Using Interceptors to Enhance CORBA", *IEEE Computer*, July 1999, pp. 62-68.

[11] *Portable Interceptors: Joint Revised Submission*, orbos/99-12-02, Dec. 1999.

[12] *Portable Interceptors: Request for Proposal*, orbos/98-09-11, Dec. 1998.

[13] S. Maffeis, "Adding Group Communication and Fault-Tolerance to CORBA", *Proceedings of the 1995 USENIX Conf. on Object-Oriented Technologies*. Monterey, CA, June 1995.

[14] *The Common Object Request Broker: Architecture and Specification, Rev. 2.3*, formal/98-12-01, June 1999.