소 특 집

통신시스템용 내장형 CORBA 기술 소개

장 종 현*, 최 원 혁*, 이 동 길*, 최 완*, 한 치 문**, 장 익 현***

*한국전자통신연구원, **한국외국어대학교 전자공학과, ***동국대학교 정보통신공학과

요 약

본 논문에서는 내장형 구조의 통신시스템에 적용할 수 있는 CORBA를 개발하기 위한 방법을 소개한다. 첫째, 통신시스템용 개발언어인 SDL과 CORBA 기반 시스템을 통합하기 위한 SDL 시스템의 통신 프로토콜과 CORBA 통신 프로토콜간의 변환 인터페이스에 대한 방법을 제시한다. 둘째, CORBA의 성능을 최적화하기 위한 객체중개자와 동일 호스트상에서 메시지 전달 오버헤드를 최적화하기 위해 공유 메모리 기반의 연동 프로토콜을 제시한다. 셋째, CORBA 기반의 통신시스템용 응용 프로그램 개발에 적합한 서비스에 대하여 설명한다. 본 모델을 기반으로 통신시스템용 SW플랫폼 하부에 CORBA를 채용할 수 있어 통신시스템의 내부 및 외부에서 분산 처리를 위한 공통 플랫폼으로 사용할 수 있다.

I. 서 론

오늘날 네트워크 구조는 인터넷의 급격한 발전과 서비스 이용자의 증가로 새로운 서비스 요구에 즉시 대응할 수 있는 형태로 발전하고 있다. 따라서 네트워크 시스템 구조는 표준 인터페이스를 이용하여 이기종간 상호 연동을 통해 새로운 서비스를 제공할 수 있는 미들웨어 기반의 개방형 구조를 지향하고 있다.

CORBA[1]는 하드웨어와 소프트웨어에 독립적인 객체지향 언어를 기반으로 하는 소프트웨어 서비스 제공 목적으로 OMG(Object Management Group)에 의해 표준 인터페이스 규격을 정의하고 있다. OMG(Object Management Group)는 1991년에 시작되었으며, 현재 800개 이상의 산업체가 참여하고 있다. 현재 개발된 다른 유형의 미들웨어로는 SUN Microsystems의 Java 언어 기반 RMI(Remote Method Invocations), Microsoft의 DCOM(Distributed Component Object Model) 및 DEC의 RPC 등이 있다. 또한 CORBA를 이용한 분산 플랫폼은 일반적인 응용에서부터 실시간 통신시스템 및 내장형 시스템으로 확장되어 사용되고 있다.

본 논문에서는 CORBA 관련 분야의 연구 내용을 분석하고, 통신 시스템용 내장형 CORBA 설계 및 구현 기술에 대하여 소개한다. 그리고 앞에서 언급한 기능을 개발하고 있는 uniORB[6]를 소개하고 이를 통해 성능을 분석한다. 또한, uniORB를 기반으로 구현된 플랫폼의 적용 사례를 기술하고자 한다.

II. 관련 연구 분석

1. CORBA

CORBA는 800개 이상의 산업체가 참여하여 구현이 아닌 인터페이스를 정의하는 규격으로 주요 특징으로는 객체 위치 투명성, 연결 및 메모리 관리, 매개 변수 (언)마셜링, 이벤트 및 요구의

(역)다중화, 오류 처리 및 고장 감내, 객체/서버의 활성화, 병행성 및 보안 기능을 제공한다. 따라서 CORBA는 언어, 운영체제, 하드웨어 및 네트워크 프로토콜의 이질적인 의존성으로부터 독립적인 응용의 구현할 수 있는 인터페이스를 제공한다[1,2].

CORBA 표준에서는 시스템 개발을 위한 다음과 같이 4가지의 인터페이스를 정의하고 있다.

1) IDL(Interface Description Language)

구현 객체에 대한 인터페이스는 프로그램 언어에 따라서 매핑을 서로 달리하는데, IDL 컴파일러는 객체에 대한 인터페이스를 표현하기 위하여 프로그래밍 언어에 독립적이며, 생성된 코드는 구현 객체에 대한 서비스를 투명하게 제공할 수 있는 수단을 제공하여야 한다.

2) ORB(Object Request Broker)

CORBA를 이용한 클라이언트 프로그램은 객체 서비스 중개자를 통하여 동일 시스템 또는 원격 시스템에 위치한 서버측 구현 코드를 투명하게 요구를 할 수 있는데, 객체 서비스 중개자는 클라이언트의 서비스 요구를 가로채어, 객체 서비스를 제공하게 될 객체 어댑터를 찾아서 매개변수 값을 포함하여 해당 메쏘드를 호출한 후 반환 값이 필요한 경우 결과 값을 반환 받는 절차를 수행한다. 이때 클라이언트는 서비스 구현 객체의 위치, 운영체제 및 하드웨어 및 프로그래밍 언어에 대한 정보는 알 필요 없이 서비스를 요구하게 된다. 이와 같이 객체 서비스 중개자는 이질적인 시스템에 구현된 구현 객체에 대한 서비스를 제공할 수 있는 위치 투명성과, 프로그래밍 언어 독립성 및 다중의 구현 시스템을 지원할 수 있는 상호 운용성을 제공한다.

3) POA(Portable Object Adaptor)

객체 어댑터는 객체 서비스 중개자와 구현 객체 사이의 매개체 역할을 담당하며, 구현 객체의 활성화, 서비스 요구 및 객체에 대한 상태 정보를 유지하면서 서비스를 제어한다.

4) IOP(Interoperability ORB Protocol)

일반적으로 ORB간 통신 프로토콜로는 TCP/IP 프로토콜 기반으로 일반적인 상호 연동 규격을 정의한 GIOP(General Inter-ORB Protocol)와 IIOP(Internet Inter-ORB Protocol)을 사용하여 통신하게 되는데, OMG에서는 전달 계층 프로토콜로 특정 프로토콜을 정의하고 있지 않다. GIOP는 통신은 서비스 요구측에서 자신의 하드웨어 구조 정보, 매개 변수 값을 포함하는 정보를 마셜링/언마셜링 하기 위한 규격을 정의한다.

2. CORBA 제품 기능 비교

현재 내장형 통신시스템에 적용할 수 있는 CORBA로는 루슨트에서 개발된 omniORB[9], 프랑크프루트 대학을 중심으로 개발되는 MICO[8] 및 워싱턴 대학에서 개발되고 있는 실시간 CORBA 규격을 준수하는 TAO[3,5]가 있다. 그리고 상용 제품으로는 미국 Vertel사에서 개발된 e*ORB[7]가 그 대표적이라 할 수 있다. 이들 제품들의 기능을 비교해 보면 다음 〈표 1〉과 같다.

3. SDL(Specification and Description Language)

SDL[10,11]은 통신 시스템의 개발 시간과 절감과 시스템의 질적 향상을 위해 ITU-T(International Telecommunication Union-Telecommunication Standardization Sector)가 권고한 시스템의 명세(specification)와 기술(description) 언어이다. 이 언어는 유용성을 입증되어 70년대 중반부터 통신 산업뿐만 아니라 교환기 시스템과 같은 실시간 수행을 필요로 하는 모든 영역에서 널리 쓰이고 있다.

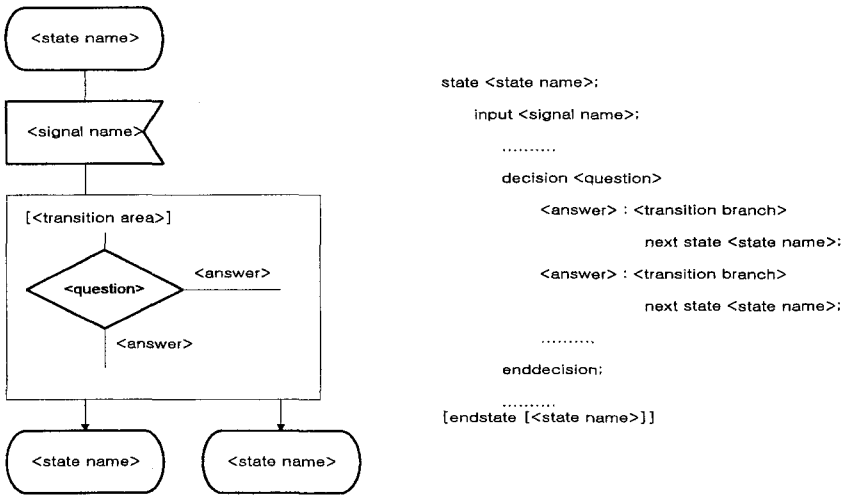
〈그림 1〉과 같이, SDL은 시스템의 명세와 기술을 위해 흐름도 형태의 그래픽 다이어그램으로 전체 시스템을 표현한다. SDL은 시스템의 구조와 행위를 명확하게 보여주고, 제어와 자료의 흐름을 시각화 하므로 유한 상태 기계에 의해 효과적으로 표현되는 통신 시스템의 명세와 설계에 적합하다.

〈표 1〉 CORBA 제품 기능 비교

구 분	Features	Services	IOP
OmniORB 2.8.1	CORBA 2.0 compliant multi-threaded ORB with complete C++ language mapping IIOP as native protocol No Repository **GNU General Public License** OmniORB2+Omnithread+OmniParTcl	**Naming Service** Lifecycle Dynamic Thread CallBack	GIOP1.0 IIOP1.0
MICO 2.3.1	CORBA 2.2 compliant supports the IDL to C++ mapping, DSI/DII, IIOP, full BOA implementation all activation modes, support for object migration and the implementation repository GUI-based front-end in Java to the MICO-ORB GNU public license	**Naming Service** **Dynamic** **Event** Trader Property Relationship A/V Streams Time	GIOP1.2 IIOP1.1
TAO	CORBA 2.2 compliant ORB Endsystem Architecture network interface operating system Repository communication protocol CORBA middleware mechanisms high-performance real-time ORB optimized IIOP engine **freely available**	Naming Trader **real-time Event Service** **Concurrency** **Time** Property A/V Streams **Logging** LifeCycle **Nodification** **Scheduling**	GIOP1.1 IIOP1.1
e*ORB	. **"Minimal" & "real-time" CORBA specs** **Telecom services** Scalable Architecture Multi-Threaded Environment Distribution Heterogenity	Recovery, Replication Monitoring, Trader **Notification**, Security Logging, Mobility Streams, **Scheduling** Time, Lifecycle **realtime**	GIOP1.2 IIOP1.1

*Bold style : 통신시스템에 적용되는 서비스

따라서, SDL을 통한 시스템의 명세는 명세 및 기술 문서로부터 소프트웨어의 정형적인 검증과 확인을 가능하게 하고, 이를 바탕으로 구현 프로그램의 자동 생성을 가능하게 한다. 또한, 시스템의 행위가 통신을 수행하는 확장된 유한 상태 기계(Communicating Extended Finite State Machine)형태로 기술되므로 프로세스 간의 상호 작용이 중점이 되는 통신 및 실시간 분산 시스템을 설계하고 시험하는데 적합하다. 이러한 이유로 실제적인 산업 분야에서는 SDL을 지원하는 도구들이 개발되어 사용되고 있다.



〈그림 1〉 SDL GR과 PR의 예

Ⅲ. 통신시스템용 내장형 CORBA 설계

미들웨어에서 성능 향상을 위하여 객체 서비스 중개자가 통신 및 요구 수행을 어떻게 병행성을 제공할 수 있느냐 하는 방법을 말하는데, 단일 쓰레드와 멀티 쓰레드 형태의 두 가지 병행 모델을 생각할 수 있다. 단일 쓰레드 모델은 하나의 쓰레드 환경에서 클라이언트로부터 요구, 실행 및 응답의 단계를 수행하지만, 멀티 쓰레드 모델은 객체 서비스 중개자가 다중의 쓰레드, 즉 쓰레드 풀을 사용하여 서비스를 처리하게 된다. 본 논문은 우선 순위 기반의 쓰레드 풀을 이용한 객체 서비스 중개자("이하 uniORB라 한다"), 고성능 연동프로토콜 및 IDL 컴파일러의 확장 기능 설계 및 구현에 있다.

1. 멀티쓰레드 객체 서비스 중개자 (Object Broker)

멀티쓰레드 기반 미들웨어란 서버측의 쓰레드 처리기의 제어에 의하여 서비스 요구를 처리하게 되는데, 이 모델의 병행 서비스 요구를 제어할 수 있고 쓰레드를 미리 생성 및 재사용이 가능하며 휴지상태의(idle) 쓰레드의 우선순위를 관리할 수 있는 장점이 있지만, 중첩된 메쏘드 기동을 허용하는 콜백을 이용한 응용에서는 Deadlock이 발생할 수 있다.

멀티쓰레드 기반 미들웨어의 성능 최적화를 위하여 서버 시스템의 운영체제, 하드웨어 제원 및 시스템 부하에 따라 고정 쓰레드 생성 및 소멸 기능을 제공할 수 있는 쓰레드 풀 기반의 시스템을 설계하였다.

일반적인 미들웨어에서 병행성 제공 모델은 다음과 같다.

1) Thread per Object

객체 서비스 중개자는 구현 객체 식별자(Object Id)를 이용하여 서번트 객체 값의 키 값을 기반으로 쓰레드를 생성하여 서비스 요구를 처리하게 된다.

2) Thread per POA

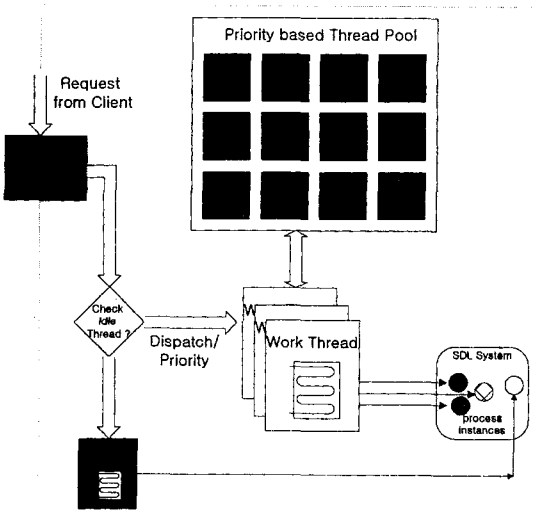
서비스 응용에 따라 구분된 Child-POA를 생성하여 서비스를 객체 활성화 정책을 관리할 수 있는데, 이 모델은 POA에 따라서 쓰레드를 생성하여 구현 객체를 실행할 수 있게 되는데, 객체 어댑터의 수가 아주 많은 경우 클라이언트의 서비스 요구에서 구현 객체에 대한 해당 객체 어댑터를 찾는데 상당한 오버헤드가 발생할 수 있다.

3) Thread per Request

객체 서비스 중개자는 클라이언트로부터 각각의 요구에 대하여 쓰레드를 생성하므로 서비스 요구가 지연되지 않고 서번트 객체를 실행하게 되고, 수행이 완료되면 쓰레드를 소멸하게 된다.

쓰레드 풀 모델은 클라이언트의 요구를 처리하기 위한 쓰레드 풀을 사용하게 되는데, 쓰레드는 단지 한번 생성되며 다른 요구를 처리하기 위하여 재사용하게 된다. 이 모델은 다중 프로세서로 구성된 시스템에서 효과적이며, 서비스 요구가 아주 빈번하게 발생하는 응용에서는 처음 생성하게 될 쓰레드의 수를 시스템의 성능에 맞게 생성함으로써 쓰레드 생성에 소비되는 시간을 없애므로 최상의 성능을 얻을 수 있지만, 제한된 쓰레드 이상의 요구가 발생하는 경우 처리의 지연이 발생할 수 있다.

따라서 본 플랫폼에 적용된 구현 모델은 〈그림 2〉와 같이 고정된 쓰레드를 이용한 쓰레드 풀 방식과 동적으로 시스템의 부하를 평가하여 서비스 지연을 방지할 수 있도록 동적 쓰레드 방식을 병행하며, 클라이언트로부터 전파된 우선순위를 기반으로 객체 어댑터 활성화의 병행성 모델을 제공할 수 있도록 한다. 또한 응용에 따라서 단일 쓰레드 모델을 이용하여 쓰레드 생성에 따른 오버헤드를 줄일 수 있도록 하였다.



〈그림 2〉 멀티 쓰레드 기반 미들웨어 구현 모델

또한 uniORB에서는 동일 플랫폼에서는 최상의 성능을 제공하기 위하여 상호 연동 프로토콜 규격을 최적화하고, 다른 범용 미들웨어의 상호 연동은 OMG GIOP 규격을 준수하도록 설계하였다. 이러한 방식은 서버로부터 클라이언트측으로 객체 참조 정보 내에 프로토콜을 포함하여 분배하며 클라이언트는 선택된 프로토콜을 이용하여 연결을 요청하여 프로토콜을 설정하도록 한다.

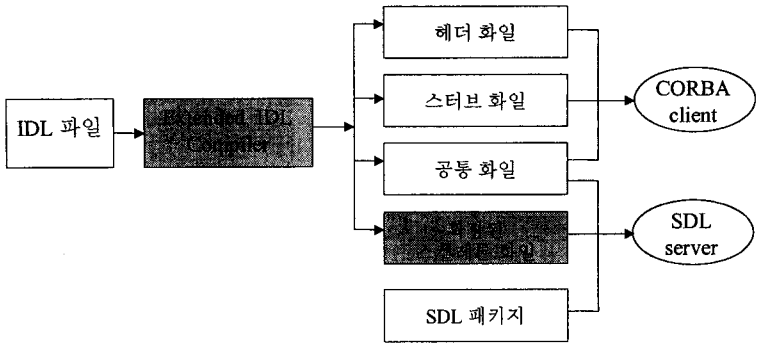
2. 고성능 연동프로토콜(Inter-Operability Protocol)

인터넷 연동 프로토콜(IIOP)은 TCP/IP 기반의 네트워크 상에서 여러 개의 객체 중개자(ORB) 간의 통신을 위해 표준 메시지 형식과 공통 데이터 표현형의 세트로 정의된 GIOP 메시지를 교환하는 프로토콜을 의미한다. 일반적인 IIOP 프로토콜은 TCP/IP를 기반으로 하지만, 시스템 내부의 객체간 통신을 위해서는 유닉스 도메인 소켓을 이용하기도 한다.

그러나 통신 시스템용 미들웨어 플랫폼의 실시간, 고성능의 요구사항을 만족시키기 위해 시스템 내부 객체간 통신을 위한 공유 메모리 기반의 IIOP 프로토콜 구현이 필수적이다. 즉, 동일 호스트상에서 공유 메모리를 사용하여 GIOP 메시지를 전달하게 되면 메시지 전송 오버헤드를 줄일 수 있다. 따라서 미들웨어 플랫폼의 성능을 향상시킬 수 있는 공유 메모리 기반의 인터넷 연동 프로토콜을 이용한 최적화 방법을 도출할 수 있다.

3. IDL 컴파일러 확장

IDL 컴파일러는 〈그림 3〉과 같은 단계로 실행되며, 최상의 성능을 위하여 고정된 우선 순위의 객체 어댑터(POA) 쓰레드를 생성하고, 클라이언트로부터 전달되는 우선 순위와 마이크로 초 단위의 타임아웃 시간의 제약을 하나의 with절을 IDL인터페이스의 각 메쏘드 마다 속성을 정의하여 하나의 인터페이스에 메쏘드와 속성의 집합 형태로 사용할 수 있도록 IDL 컴파일러를 구현하였다. 또한 SDL 시스템 인터페이스를 위하



〈그림 3〉 IDL 컴파일러 실행 모델

여 시스템으로의 시그널을 전달하기 위하여 xGetSignal(objectname) 프리미티브를 통하여 해당 객체에 대한 시그널 정보 구조체(yOutputSignal)를 할당하고, 클라이언트로부터 전달된 매개 변수 값을 시그널정보 구조체의 각각의 매개 변수 값으로 매핑시킨다.

이와 같이 생성된 시그널 정보를 SDL커널을 통한 시스템과의 통신 프리미티브인 SDL_Output(yOutputSignal)을 이용하여 전달하고 응답이 필요한 경우 SDL 시스템으로부터 쓰레드 조건 시그널 응답을 기다리게 된다. SDL 시스템은 객체에 대한 동작을 수행한 후 응답이 결과 시그널을 처리할 수 없을 경우 외부 블록과의 통신을 위하여 시그널 출력 환경 함수를 통하여 스켈레톤으로 신호를 전달하게 된다. 여기서 시스템의 부하를 최소화할 수 있는 기법을 적용하여야 한다.

Ⅳ. 서비스

1. 네이밍서비스

네이밍 서비스는 이름을 이용하여 CORBA 객체가 다른 객체를 찾을 수 있도록 지원하는 서비스로 하나의 객체에 여러 개의 논리적인 이름을 연관시킬 수 있도록 해 준다. 먼저 이름과 객체를 연결 짓는 것을 바인딩이라 하는데, 어떤 객체를 바인딩하면 네이밍 컨텍스트내에 이름과 객체에 대한 정보를 그래프 형태로 구성하게 된다. 클라이언트가 이름을 매개변수로 해석을 요구하면 네이밍 서버는 이 그래프를 탐색함으로써 복합 이름(compound name)을 이용하는 특정 객체를 찾아 객체 정보를 반환하게 된다.

다중의 클라이언트가 하나의 제어 보드에 의해 동작하는 구현 객체로 접근할 경우 네이밍 서비스를 이용한다. 이때 일반적으로 네이밍 서비스가 동작하는 보드의 하드웨어 불안 또는 사용자에 의해 재시동하는 경우 서버 객체를 재 실행하여야 서비스를 요청할 수 있다.

2. 타이머 관리 서비스

유닉스 기반의 범용 운영체제 환경에서 실시간 운영체제가 제공하는 실시간 특성을 제공하기는 쉽지 않다. 따라서, 본 논문에서는 실시간 CORBA 규격에서 시간 제어 부분으로 서버로부터 무응답을 방지하기 위한 타임아웃 기능, 특정 시간에 임의의 객체에 접근할 수 있는 상대/절대시간 서비스 요구 및 서비스의 전달 지연과 응답 지체와 같은 제한적인 실시간 특성을 지원한다.

타임아웃 서비스는 클라이언트측에서 동기식 블럭킹 모드의 서비스 요구 단점을 보완하기 위해, 서버의 서비스 응답을 제한된 시간 동안 기다린 후 응답이 없는 경우를 대비하여 시스템의 서비스 연속성을 제공하기 위함이다. 이 서비스는 IDL 인터페이스 정의 파일에 다음과 같은 형태로 제한된 시간을 정의하면, IDL 컴파일러에서 해당 기능을 처리할 수 있는 코드를 생성한다.

long ReturnedTimeout(in long first, in long second) with (timeout=2000000);

스터브 코드에서 서버측으로 서비스를 요청한 후 timeout 시간 동안 응답이 없을 경우 오류 타입을 CORBA_TIMEOUT_SERVER로 정의하고, 할당된 입출력 버퍼를 해제하고 반환 값을 "0"으로 정의하여 제어를 객체 서비스 중개자로 넘긴다.

서비스 전달 지연 및 응답 지체 서비스는 프로세스간 동기화를 목적으로 사용될 수 있으며 IDL 인터페이스 정의는 다음과 같다.

long delayedReturn(in long input) with (delayed_time=1000000);
long deferedRequest(in long input) with (defered_time=1000000);

상대/절대 시간 서비스 요구 기능은 현재 시간으로부터 분, 시간과 같이 비교적 긴 시간이 지난 후에 객체에 접근할 수 있도록 하는 기법으로 주로 멀티쓰레드 기반 응용 프로그램의 콜백 함수를 활용한다.

long atTimeReq(in long input) with(at=200207091955);/ 2002년 7월 9일 19:55 */*
long atTimeReq(in long input) with(at=+3600);/ 현재 시간으로부터 1시간 후 */*

V. 성능 분석

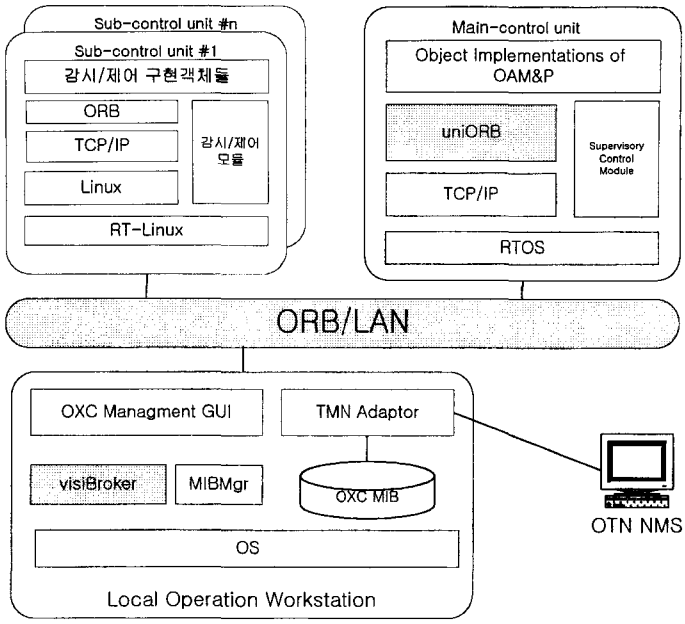
성능 분석은 아무런 동작을 수행하지 않고 함수 반환 기능만을 수행하는 서버 프로그램에 대하여 OMG 규격에 정의되는 다양한 형태의 데이터 타입에 대하여 일정한 횟수를 호출하는데 소요된 시간을 측정하여 초당 처리율을 비교하였다. 성능 분석 시스템은 썬사의 솔라리스 버전 2.7 운영체제와 UltraSparc 166MHz, 256MB 바이트의 주 기억장치의 시스템에 uniORB와 독일에서 공개용으로 개발된 MICO(Version2.3.5) 및 AT&T Technologies에서 개발된 omniORB (3.0.4) 3가지 미들웨어를 대상으로 비교하였다.

단일 클라이언트에서 동일한 함수를 1000회 호출하는 시험에서 〈그림 8〉에서 보는 바와 같이 본 논문에서 개발된 uniORB는 MICO에 비하여 약 2.5배, omniORB에 비하여 약 20%의 우수한 성능을 보였다.

또한, 5개의 다중 클라이언트를 이용하여 성능을 측정한 결과 〈그림 9〉와 같이 uniORB는 MICO에 비하여 각각의 데이터 타입에 대하여 약 3.5배, omniORB에 비하여 2.5배 이상을 나타내었다. 따라서, TCP/IP 기반의 IIOP 프로토콜을 최적화하여 구현함으로써 다중의 클라이언트의 요구 처리에서 상대적으로 낮은 성능 저하를 확인할 수 있었다.

VI. 결 론

본 논문에서 제시한 기술들을 uniORB에 구현하였으며 uniORB는 현재 한국전자통신연구원에서 개발 중인 OXC(Optical Cross Connector) 시스템[12]의 하부 통신 플랫폼으로 사용되고 있다. OXC(Optical Cross Connector) 관리시스템은 〈그림 4〉와 같이 OXC 운용관리를 위한 분산관리 플랫폼으로 개발되었으며, 사용자 정합 장치와 운용 관리 시스템간은 상용 CORBA 제품인 VisiBroker와 uniORB의 연동을 통하여 객체를 액세스하고 있음을 보여준다. 각각의 제어 보드들은 객체지향 모델링 과정에서 만들어진 OXC 운용관리객체를 수행할 수 있도록 설계하였으며, OXC 운용 관리 시스템은 시스템 운영자를 위한 GUI 환경을 제공한다. 특히, 본 응용에서는 여러 개의 보드로 구성된 서버 제어장치에 위치한 객체 서비스에 본 논문에서 구현된 네이밍 서비스를 적용함으로써 장치의 탈/부착에 의한 이중화에서 서버 제어장치의 재 구동 없이 서비스를 계속할 수 있는 신뢰성을 제공하게 되었다.



〈그림 4〉 OXC의 OA&M 관리 플랫폼 구조

OXC 시스템에서 uniORB를 적용한 결과 uniORB는 통신시스템 내부에서의 통신 플랫폼으로 사용하는데 성능, 실시간 처리, 통신, 신뢰성 등에 문제가 없음을 보여 주었다. 향후 uniORB를 내장형 시스템에서 실행되는 통신용 소프트웨어에 더욱 더 폭 넓게 적용하기 위해서 통신시스템을 위한 실시간 처리 기능의 추가 개발, 메모리 사용의 최적화, 시스템의 안정화, 고장감내 등의 관련 연구를 추진할 계획이다.

참 고 문 헌

〔1〕 Object Management Group, *The Common Object Request Broker : architecture and Specification*, 2.4 ed., Oct. 2000.

〔2〕 *Object Management Group, Realtime CORBA Joint Revised Submission*, OMG Document orbos/99-02-12 ed., March 1999.

〔3〕 Schmidt and F. Kuhns, "An Overview of the Real-time CORBA Specification," *IEEE Computer Magazine, Special Issue on Object-oriented Real-time Computing*, June 2000.

〔4〕 Using SDL to develop CORBA object implementations, Morgan Bjkander, Telelogic AB, 1997.

〔5〕 C. Schmidt, D. L. Levine, and S. Mungee, "The Design and Performance of Real-Time Object Request Brokers," *Computer Communications*, vol. 21, Apr. 1998.

〔6〕 개방형S/W플랫폼팀, "uniORB 사용자 매뉴얼,", ETRI, Dec, 2001

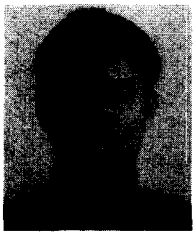
〔7〕 *Ann Marie O'Connor*, "Project e*ORB-CORBA in Telecommunications,", 2000

〔8〕 Puder and K. Römer, "MICO-User and Reference Manual. Technical Report TR-98-031, International Computer Science Institute, Berkeley.

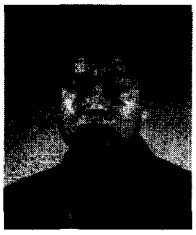
〔9〕 Sai-Lai Ro, David Riddoch and Duncan Grisby, "The omniORB version 3.0 User's Guide,", AT&T Laborator-

ies Cambridge, May 2000
〔10〕 Using SDL to develop CORBA object implementations, Morgan Bjkander, Telelogic AB, 1997.
〔11〕 The CORBA Integration, SDT manual, 1997.
〔12〕 박수명, 이상화, 남현순, 주성순, "CORBA 기반의 OXC 운용 관리 플랫폼 설계," Optoelectronics and Microelectronics 2001, Nov. 2001

저 자 소 개

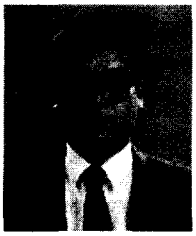


張 鍾 鉉
1988년 경북대학교 전자공학과(공학사), 2000년 충남대학교 전자공학과(공학석사), 현재 한국외국어대학교 전자공학과 박사과정, 1988년 2월~1994년 2월 : 대우통신(주) 종합연구소, 1994년 3월~현재 : 한국전자통신연구원 교환전송기술연구소 선임연구원, 〈주관심 분야 : 네트워크 보안, 이동통신, 미들웨어 등〉



崔 元 赫
1999년 2월 경북대 컴퓨터공학과 졸업, 2001년 2월 경북대 컴퓨터공학과 석사, 2001년 2월~현재 : 한국전자통신연구원 연구원, 네트워크S/W플랫폼팀 연구원, 〈주관심 분야 : 소프트웨어 공학, 미들웨어〉

李 東 吉
1983년 경북대학교 전자공학과(공학사), 1985년 한국과학기술원 전산학석사, 1994년 한국과학기술원 전산학박사, 1985년~현재 : 한국전자통신연구원 책임연구원, 〈주관심 분야 : 컴파일러 구성론, 프로그램밍 언어론, 미들웨어 등〉



崔 完
1981년 경북대학교 전자공학과(전자계산전공)(공학사), 1983년 KAIST 전산학과(공학석사), 1983년 3월~1985년 2월 : KAIST 전산학과 조교, 1985년 3월~현재 : 한국전자통신연구원 책연, 팀장, 1988년 8월 : 정보처리(전자계산기조직응용) 기술사, 2000년 4 월 SPICE 심사원, 〈주관심 분야 : 실시간 시스템 OS/DBMS/Middleware, 통신프로토콜, S/W 공학〉

韓 致 文
1977년 경북대학교 전자공학과(공학사), 1983년 KAIST 전산학과(공학석사), 1990년 The University of Tokyo(동경대) 전자정보공학과(공학박사), 1977년 2월~1983년 3월 : 한국과학기술연구원(KIST) 연구원, 1983년 4월~1997년 2월 : 한국전자통신연구원(ETRI) 선연, 책연 교환기술연구단 계통연구부장 역임, 1997년 3월~현재 : 한국외국어대학교 전자공학과 교수, 〈주관심 분야 : ATM 통신망 및 교환, IMT-2000 네트워크, 네트워크 보안 등〉

張 益 鉉
1984년 서울대학교 계산통계학과(이학사), 1986년 한국과학기술원 전산학석사, 1998년 한국과학기술원 전산학박사, 1986년 2월~1999년 2월 : 데이콤(주) 책연, 1999년 3월~현재 : 동국대학교 정보통신공학과 조교수, 〈주관심 분야 : 컴퓨터통신, 컴퓨터네트워크, 분산시스템, 병렬처리시스템 등〉