개방형 네트워크 환경을 위한 멀티쓰레드 기반 코바 설계 및 구현

장 종 현[†] · 이 동 길[††] · 한 치 문[†††]

요 약

분산컴퓨팅 시스템은 서로 이질적인 시스템간 상호 연동성 문제를 해결하기 위한 새로운 시스템 아키텍처를 제시한다. 본 논문에서는 개방형 통신시스템에서 물리적으로 분리된 시스템에서 실행되는 소프트웨어 블록간 분산 처리를 위한 멀티쓰레드 기반의 통신 시스템용 미들웨어를 개발하기 위한 요구 기능 분석, 프로토콜 구현 및 벤치마킹을 통한 시스템의 성능을 분석을 통한 최적의 미들웨어 플랫폼 구현에 활용하고자 한다. 통신 시스템용 미들웨어의 주요 기능으로는 제한적인 서비스 품질(QoS)을 제공하기 위한 우선순위 기반의 서비스 처리 및 타임아웃 기능과 예외 상황이 발생하는 경우 관련 블록으로 상태 정보를 통보할 수 있는 통지(Notification) 서비스의 제공이 필수 요구 조건이라 할 수 있다. 아울러, 최적의 성능을 만족할 수 있는 고속의 멀티 쓰레드 기반으로 확장성, 유연성 및 플랫폼의 견고성이 요구되는 미들웨어의 설계 및 구현에 그 목적을 두고 있다.

Design and Implementation of CORBA based on Multi-Threaded in Open Network Environments

Jong Hyun Jang[†] · Dong Gill Lee[††] · Chi Moon Han[†††]

ABSTRACT

Distributed computing system gives a new system architecture to be taken into consideration for solving the problems of interoperability of heterogeneous systems. In the present paper, CORBA based on multi-threaded interoperates with software blocks at physically isolated hardware. We show how archives optimal CORBA system from analysis of required functions, implementations of protocols and benchmarking of system performance in the Open Multi-service Network System Environment. The core features of our CORBA system are restricted Quality of Service based on priority, timeout service and exception status information notify to the related software blocks. And the objectives are design and implementation of high performance multi-threaded middleware and satisfied with extendibility, flexibility and stability of CORBA platform.

키워드 : 코바(CORBA), 미들웨어(Middleware), 개방형통신시스템(MSF), SDL

1. 개 요

CORBA는 하드웨어와 소프트웨어에 독립적인 객체지향 언어를 기반으로 하는 소프트웨어 버스를 제공하기 위하여 OMG (Object Management Group) 의해 1991년에 시작되어 현재 800개 이상의 산업체가 참여하여 구현이 아닌 표준 인터페이스 규격을 정의하고 있다. 현재 개발된 다른 미들웨어로는 Java언어 기반의 RMI(Remote Method Invocations), 마이크로소프트웨어사에서 개발된 DCOM(Distributed Component Object Model)및 DEC의 RPC가 있으나, 현재는 RMI, DCOM 및 CORBA를 이용한 분산 응용 프로그램을 개발하고 있으며, CORBA를 이용한 분산플랫폼은 일반적인 응용에서부터 실시간 통신 시스템 및 내장형 시스템으로 확장되고 있다.

또한 현재 OMG에서는 규격화되고 있는 실시간 CORBA는 기존 코바의 기능에서 QoS 제어 기능을 제공하기 위한 우선순위 역전 방지 및 자원 관리 기능을 기반으로 응답 예측성을 제공하고자 한다. 실시간 특성을 제공하기 위하여서는 우선 순위 정책 기반의 프로세스, 통신 및 메모리 자원에 대한 제원 및 제어할 수 있어야 한다. 오늘날 급속한 기술 개발로 하드웨어는 점점 작고, 빠르고, 싸지고 있는 반면에, 소프트웨어는 점차 대형화에 따라 점점 느려지고 있지만 보다 값비싸져 분산시스템의 구축이 어려운 실정이다. 따라서, 대규모 개방형 스위칭 시스템, 항공 통제와 같은 특수 목적의 분산 어플리케이션들은 실시간 QoS의 보장이 요구되고 있지

† 정 회 원 : 한국외국어대학교 대학원 전자공학과
†† 정 회 원 : 한국전자통신연구원 책임연구원
††† 정 회 원 : 한국외국어대학교 전자공학과 교수
논문접수 : 2001년 7월 23일, 심사완료 : 2001년 11월 22일

만, 기존의 미들웨어들은 효율적인 실시간 QoS 요구사항을 지원하고 있지 못한다. 보다 효과적인 실시간 QoS를 제공하는 분산 처리를 위하여 다중 QoS 특성의 동시 제어를 위한 실시간 내장형 미들웨어 기술을 이용한 최적의 성능 및 확장성을 제공하고 일반적인 기술 기반의 고객화된 구성을 제공할 수 있는 미들웨어 기술을 통한 표준화를 진행하여야 한다.

본 논문에서는 개방형 통신시스템 개발에서 물리적으로 분리된 시스템에서 실행되는 소프트웨어 기능 블록간 분산 처리를 위한 멀티쓰레드를 기반으로 개방형 통신 시스템용 미들웨어 개발하기 위한 요구 기능 분석, 프로토콜 구현 및 벤치마킹을 통한 시스템 성능 분석으로 최적의 미들웨어 플랫폼 구현에 활용하고자 한다.

통신 시스템용 미들웨어의 주요 기능으로는 제한적인 서비스 품질(QoS)을 제공하기 위한 우선순위 기반의 서비스 처리 및 타임아웃 기능과 예외 상황이 발생하는 경우 관련 블록으로 상태 정보를 통보할 수 있는 통지(Notification) 서비스의 제공이 필수 요구 조건이라 할 수 있다. 아울러, 최적의 성능을 만족할 수 있는 고속의 멀티 쓰레드 기반으로 확장성, 유연성 및 플랫폼의 견고성이 요구되는 미들웨어의 설계 및 구현에 그 목적이 있다.

2. 미들웨어(CORBA) 규격

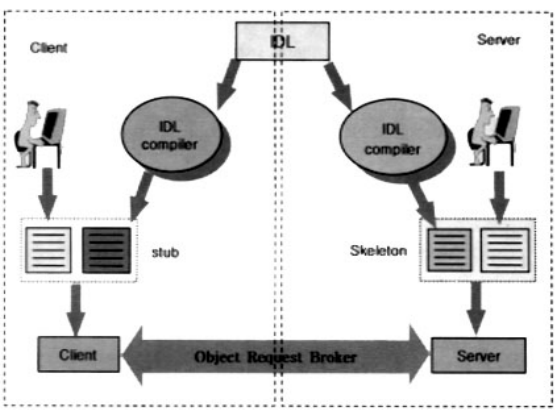
2.1 CORBA 표준

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

- 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.
- 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 매개 변수 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.
- POA(Portable Object Adaptor)
  객체 어댑터는 객체 서비스 중개자와 구현 객체사이의 매개체 역할을 담당하며, 구현 객체의 활성화, 서비스 요구 및 객체에 대한 상태 정보를 유지하면서 서비스를 제어한다.
- 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 마샬링/언마샬링하기 위한 규격을 정의한다.

CORBA를 이용한 분산 시스템 개발 과정은 (그림 1)과 같다.



(그림 1) CORBA를 이용한 분산 시스템 개발 과정

2.2 실시간 CORBA 규격

실시간 CORBA는 기존 코바의 기능에서 QoS 제어 기능을 제공하기 위한 우선순위 역전 및 자우선 정책 기반의 프로세서, 통신 및 메모리 자원에 대한 제원 및 제어할 수 있어야 한다.

원 관리 기능을 기반으로 응답 예측성을 제공하여야 한다. 실시간 특성을 제공하기 위하여서는 오늘날 급속한 기술 개

발로 하드웨어는 점점 작고, 빠르고, 싸지고 있는 반면에, 소프트웨어는 점차 대형화에 따라 점점 느려지고 있지만 보다 값비싸져 분산시스템의 구축이 어려운 실정이다. 따라서, 대규모 개방형 스위칭 시스템, 항공 통제와 같은 특수 목적의 분산 어플리케이션들은 실시간 QoS의 보장이 요구되고 있지만, 기존의 미들웨어들은 효율적인 실시간 QoS 요구사항을 지원하고 있지 못한다. 따라서, 보다 효과적인 실시간 QoS를 제공하는 분산 처리를 위하여 다음과 같은 접근 방법의 미들웨어의 개발이 요구된다[3].

- 다중 QoS 특성의 동시 제어를 위한 실시간 내장형 미들웨어 기술을 이용한 확장성 및 최적의 성능 만족
- 일반적인 기술 기반의 고객화된 구성을 제공할 수 있는 미들웨어 기술을 통한 표준화

실시간 CORA의 규격에서는 종단간 서비스 허용 시간의 예측성의 지원을 그 목적으로 하는데, 이를 지원하기 위하여 클라이언트와 서버측간에 서비스 처리를 처리하는 동안 고정된 우선순위 기반 시스템, 우선 순위의 전도(inversion) 회피 및 서비스 지연의 한계 등을 이용하여 허용된 시간내에 응답의 예측성을 제공하여야 한다.

실시간 CORBA는 크게 다음과 같은 4가지의 주요 컴포넌트로 구성된다.

2.2.1 운영체제 기반의 스케줄러

플랫폼의 이식성을 제공하기 위하여 운영체제와 독립적인 스케줄링 기법을 제공하기 위하여 객체 중개자 내부의 스케줄러의 우선순위와 운영체제의 우선순위와의 매핑 기능과 클라이언트 정의 우선 순위를 서버측 우선 순위로 전파 및 서버측 우선 순위 정의 기능을 제공하여야 한다. 또한 우선 순위는 실행 과정에서 전도되지 않아야 하며, 쓰레드 풀 및 연결과 할당된 버퍼 등의 자원 관리 기능을 제공하여야 한다.

2.2.2 실시간 객체 서비스 중개자(ORB)

실시간 객체서비스 중개자는 실시간 메시지 호출 정보 전송, 제한된 시간내에서의 이벤트 및 예외 처리, 쓰레드의 안정화 및 병렬화, 메시지 호출의 스케줄링 및 서비스 품질을 보장할 수 있어야 한다.

2.2.3 통신 수단

통신 채널은 TCP 또는 ATM PVC와 같은 연결형 서비스를 활용함으로써 채널 설정 시간을 최소화한다

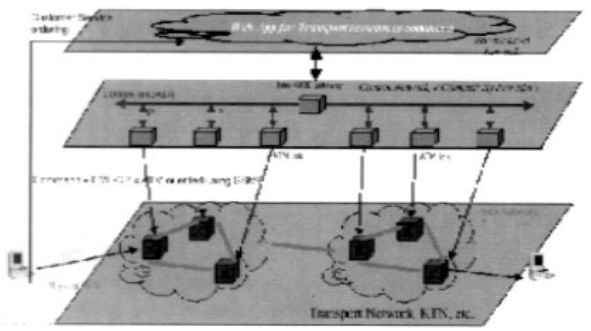
2.2.4 응용 프로그램

응용 프로그램은 실시간 특성을 활용하기 위하여서는 프로토콜, POA에 대한 우선순위 모델에 대한 선택과 쓰레드간 자원 충돌을 방지하기 위한 상호 배제 기능을 이용하여 작성하여야 한다.

3. 개방형 통신 시스템용 미들웨어 요구 기능

개방형 통신시스템은 소프트웨어와 하드웨어를 표준화된 인터페이스를 제공하는 컴포넌트로 구성하여 네트워크에 개방성을 부여하기 위함을 그 목적으로 한다. 소프트웨어 측면에서 모든 서비스 관련 소프트웨어가 표준 API를 통하여 서비스 플랫폼과 연결 구동되어, 물리적으로 분리된 플랫폼에 각각의 통신 서비스에 필요한 공통적인 기능 블록들을 두어 새로운 서비스가 등장하여도 플러그-인 개념으로 몇 개의 블록들을 조합 운용함으로써 신속하고 경제적으로 서비스가 가능해 진다[9].

개방형 통신 시스템의 주요한 특징을 살펴보면 소프트웨어 및 하드웨어의 독립적인 개발 가능, 시스템의 기능 개선, 계층적 시스템 구성 가능 및 미들웨어와 응용 서비스의 계층화와 같으며, 시스템의 구성은 (그림 2)와 같다.



(그림 2) 개방형 통신시스템의 계층적 구조[MSF Forum 자료]

서비스 제어부의 블록간 통신 기능을 제공하는 실시간 미들웨어는 크게 프로세서, 통신 채널 관리 기능을 제공하여 한다. 프로세스 관리는 유닉스 운영체제에서 제공되는 POSIX 쓰레드를 우선 순위 기반의 풀을 이용하여 제한된 실시간 기능을 제공하고, 통신 채널은 TCP 또는 ATM PVC와 같은 연결형 서비스를 활용함으로써 채널 설정 시간을 최소화한다. 범용 CORBA가 유닉스 운영체제를 기반으로 하는 시스템에서 제한적인 실시간 특성을 요구하는 개방형 통신 시스템에 적용하기 어려운 점으로는 표준화된 QoS 정책 및 기능과 실시간 응용을 지원하는 API가 존재하지 않으며 과도한 데이터 복사와 같은 지연 부하에 따른 성능을 만족하지 못한다[7, 8].

따라서 제한된 실시간 특성을 제공하기 위하여 다음과 같은 기능이 요구된다.

- 객체 위치 투명성
- 프로그래밍 언어 및 운영체제 독립성
- 다양하고 최적화된 네트워킹 환경 및 통신 프로토콜
- 우선 순위 기반의 다중 쓰레드에 의한 서비스 요구 다중화
- 최적화된 메모리 관리

4. 개방형 통신 시스템용 멀티 쓰레드 기반 미들웨어 설계 및 구현

유닉스 운영체제에서 제공되는 POSIX 쓰레드는 작업을 단일 프로세스를 이용하여 구현하기 상당히 복잡한 작업을 특정 태스크를 수행하는 전용 쓰레드에 위임하게 함으로써 응용 프로그램의 구조를 개선할 수 있다. 예를 들어 비동기 모드 블록킹 입출력, 데이터베이스 질의 또는 대규모 데이터를 정렬하는 작업에서 다중의 쓰레드를 이용함으로써 다중 프로세서 시스템인 경우 상당한 성능 향상 효과를 얻을 수 있다.

CORBA 구현 객체는 쓰레드의 사용 유무에 따라서 유닉스 프로세스와 쓰레드 타입으로 구분할 수 있는데, 프로세스 타입은 공유된 데이터 구조체를 보호하기 위한 동기화 기법을 사용할 수 없으므로 사용자에 의해 보장되어야 하며, 단지 하나의 프로세스가 운영체제에 의해 할당된 시간동안 구현 객체를 실행한다. 특히 블록킹 모드의 입출력 동작인 경우에는 오퍼레이션이 반환될 때까지 실행이 중지되므로 플랫폼의 성능을 저하시키게 된다.

그러나, 다중 쓰레드를 이용한 ORB의 구현에서는 구현 객체(서번트)를 찾고, 활성화시키는 객체 어댑터(POA/BOA)를 완전히 재진입 가능한 쓰레드 코드로 구현하며, 시스템의 제원에 따라 동적 쓰레드 생성 기법을 적용함으로써 전체 시스템의 성능 저하없이 블록킹 모드의 오퍼레이션을 실행할 수 있다.

CORBA를 사용함으로써 광범위한 호스트 시스템을 사용할 수 있으며 서로 다른 데이터 표현 기반에서 투명한 데이터 처리 기능과 IDL을 이용한 호스트 기반 프로그램이 가능하여 다양한 디바이스 및 시스템으로 쉽게 확장할 수 있다.

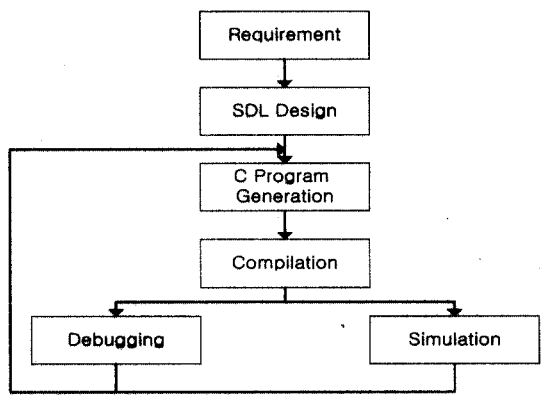
4.1 쓰레드를 이용한 응용

CORBA를 이용한 응용 프로그램의 개발은 클라이언트와 서버간의 인터페이스 정의 파일을 IDL 컴파일러를 이용하여 스터브 코드와 스켈레톤 코드를 생성하여 각각 링크하여 실행 화일을 생성함으로써 서버 시스템의 위치에 관계없이 서비스 객체를 이용할 수 있다. 클라이언트 프로그램은 단일 또는 다중의 쓰레드를 이용하여 작성할 수 있는데, 멀티 쓰레드를 이용하는 경우에는 응답이 필요한 경우 ORB내부에서 다중 쓰레드에 대하여 서비스 요구에서 응답 수신과 관련한 절차를 제어하여야 한다. 서비스 요구에 대하여서는 우선 순위 기반의 쓰레드 스케쥴링은 수행하지 않고 순차적으로 서비스를 처리하게 된다.

그러나, 서버 프로그램은 루트 POA에 대하여 서번트 객체에 대하여 활성화하고, 참조 값을 이용하여 클라이언트로부터 서비스 요구를 받을 수 있도록 IOR값을 생성한다. 그러나, 멀티 쓰레드와 관련한 기능은 ORB에서 자원 및 통신 경로를 관리하여야 한다.

4.2 개방형 통신시스템 통합 환경

개방형 통신시스템은 소프트웨어 개발방법론을 이용하여 명세 언어인 SDL(System Description Language)을 통하여 시스템을 기술하고, 자동 프로그래밍 코드 생성 툴을 이용하여 개발하는 단계를 적용하게 된다. 본 논문에서는 1980년대 말부터 확장된 상태 기계 개념을 기반으로 하여 실시간 및 내장형 통신 시스템 분야에서 설계 중심 소프트웨어 개발 도구인 SDT를 활용한 시스템 개발에 적합한 미들웨어 설계로 한정한다. SDL 통합 환경은 (그림 3)과 같이 SDL/PR을 공통의 인터페이스를 가진 자동화된 그래픽 작성기, 시뮬레이터, SDL 분석기 및 컴파일러로 구성된 통합 개발 환경이다. 소프트웨어 개발에서 구현 단계 이전에 명세 언어인 SDL을 통한 시스템의 명세는 소프트웨어 명세 및 기술 문서로부터 소프트웨어의 정형적인 검증과 확인을 가능케 하여 구현 프로그램의 자동 생성을 수행하는 것을 목적으로 한다[4, 5].



(그림 3) SDL 통합 소프트웨어 개발 환경

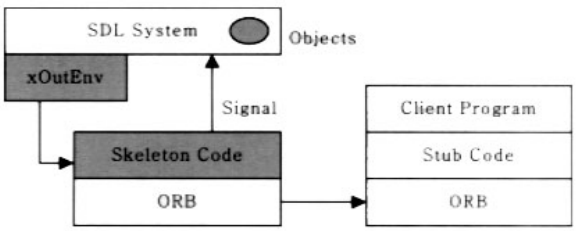
개방형 통신시스템의 개발은 SDL로 작성된 시스템 명세를 통한 자동 구현 코드 생성은 시스템의 개발 기간/비용, 소프트웨어 품질 및 코드의 재사용을 가능케 한다.

4.3 SDL과 CORBA 통합 환경

현재 개발된 SDL 통합 개발 환경 도구로는 Telelogic사에서 개발된 SDT이 있는데, 이 시스템의 구현 언어로는 C, CHILL 언어로의 변환이 가능하여, 현재 미들웨어 통합 기능은 개발되어 있지 않은 실정이다. 따라서 SDT와 CORBA 통합을 위한 일반적인 접근 방법으로는 CORBA 플랫폼을 SDL 프로세스의 실행 환경으로 이용할 수 있는데, 이 방법은 (그림 4)와 같이 SDL 시스템에서 CORBA를 위한 인터페이스를 생성한 후 IDL 컴파일러를 이용하여 SDL 구현 객체를 접근하기 위한 스켈레톤 코드를 이용하여 접근할 수 있다.

본 논문에서는 C 언어 기반의 코바의 IDL 컴파일러를 이용하여 인터페이스 파일로부터 생성된 스켈레톤 코드에서 직접 SDL 프로세스에 접근할 수 있도록 객체 시스템에 맞게 시그널을 구성하여야 한다. 이 때 스켈레톤의 매개변수를 추

출하여 SDT의 시그널 메시지를 구성하고 SDL 시스템 프리미티브를 이용하여 시그널 타입의 정보를 해당 소프트웨어 기능 블록으로 전달한다. 또한 응답은 SDT 시스템의 외부 함수 인터페이스 기능을 제공하는 환경 함수(*xOutEnv*)로부터 광역 시그널 변수 값(*GlobalSignal*)을 추출하여 클라이언트 측으로 전달하게 된다.



(그림 4) SDL CORBA 통합 환경

5. 멀티쓰레드 기반 실시간 미들웨어 설계 및 구현

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")의 설계 및 구현에 있다.

5.1 멀티쓰레드 객체 서비스 중개자

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.

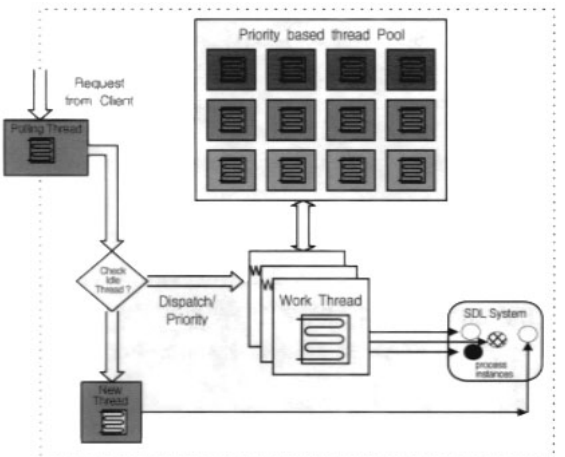
- Thread per Object
  객체 서비스 중개자는 구현 객체 식별자(Object Id)를 이용하여 서번트 객체 값의 키 값을 기반으로 쓰레드를 생성하여 서비스 요구를 처리하게 된다.

```
servant = uniORB_POA_find_servant(recv_buffer, poa, & cookie,
          &obj_impl, & oid) ;
dispatcher_pointer = & UNIORB_OBJECT_KEY(servant -> →
          _private) → dispatcher ;
```

- 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.
- 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 5)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 5) 멀티 쓰레드 기반 미들웨어 구현 모델

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

5.2 제한된 실시간 서비스 설계

5.2.1 우선 순위 기반 서비스

이질적인 운영체제 환경에서 독립적인 우선순위 기반으로 하는 광역 우선 순위 기반의 쓰레드 풀 모델을 적용하며 시스템의 요구, 부하, 제원에 따라서 우선순위를 설정할 수 있게 한다. 우선 순위에의 적용 방식은 클라이언트 우선 순위 전파 모델과 서버 정의 우선 순위 모델로 구분할 수 있는데, 전자는 클라이언트의 실행 요구 우선순위를 기반으로 스케쥴링되어 서비스가 기동되며, 후자는 동일 노드에서 다른 객체들간의 상대적 우선순위에 의하여 스케쥴링되며 특정 서버에 의하여 특성 우선 순위의 서비스를 처리할 수 있다. uniORB에서는 위의 두 가지 모델을 통합한 형태인 고정된 숫자의 우선순위를 가진 쓰레드를 미리 생성하여 쓰레드 풀을 구성하며, 클라이언트로부터 전파된 우선 순위를 객체 서비스 중개자가 해석하여 쓰레드 풀의 우선 순위를 가진 쓰레드로 서비스 실행을 할당하며, 객체 서비스 중개자 내부에서 객체들간은 아래 코드와 같이 POSIX 쓰레드 표준 스케쥴링 기법인 FIFO 방식을 적용하였다.

```
pthread_attr_setschedpolicy(&attr, SCHED_FIFO) ;
pthread_attr_setschedparam(&attr, &priority_param) ;
```

5.2.2 타임아웃 서비스

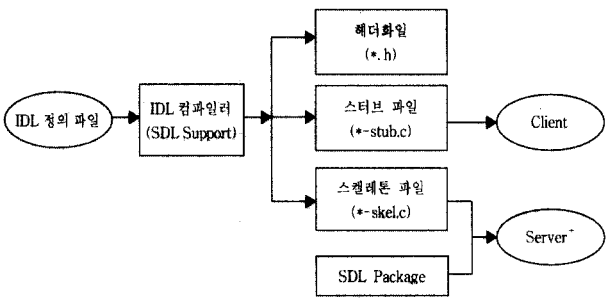
타임아웃 서비스는 클라이언트측에서 동기식 블럭킹 모드의 서비스 요구의 단점을 보완하여 서버의 서비스 응답을 제한된 시간 동안 대기 후 응답이 없는 경우 시스템의 서비스 연속성을 제공하기 위한 서비스이다. 이 서비스는 IDL 인터페이스 정의 파일에 다음과 같은 형태로 제한된 시간을 정의하면

```
long multiply (in long first, in long second) with (priority = 2,
timeout = 2) ;
```

스터브 코드에서 서버측으로 서비스를 요청한 후 timeout 시간동안 응답이 없을 경우 오류 타입을 CORBA_TIMEOUT_SERVER로 정의하고, 할당된 입출력 버퍼를 해제한 후 반환값을 "0"으로 정의한 후 제어를 객체 서비스 중개자로 넘긴다.

5.3 IDL컴파일러 설계

IDL 컴파일러는 (그림 6)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 *with*절을 IDL인터페이스의 각 메쏘드마다 속성을 정의하여 하나의 인터페이스에 메쏘드와 속성의 집합 형태로 사용할 수 있도록 IDL 컴파일러를 구현하였다. 또한 SDL 시스템 인터페이스를 위하여 시스템으로의 시그널을 전달하기 위하여 *xGetSignal(object_name)* 프리미티브를 통하여 해당 객체에 대한 시그널 정보 구조체(*yOutputSignal*)를 할당하고, 클라이언트로부터 전달된 매개 변수 값을 시그널정보 구조체의 각각의 매개 변수 값으로 매핑시킨다.



(그림 6) IDL 컴파일러 실행 모델

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.

5.4 동작

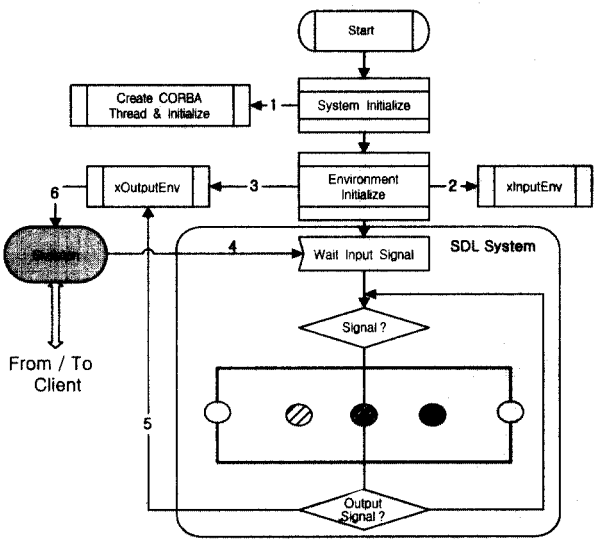
CORBA와 시그널 기반의 분산 시스템간의 통신을 위하여 SDT 개발도구로부터 혹은 프로그래머에 의해 작성된 인터페이스 정의 파일을 IDL 컴파일러를 이용하여 생성된 스터브 코드를 이용하여 클라이언트 프로그램을 생성한다. 그러나 서버측에 대하여 SDL 정합 기능이 포함된 스켈레톤 코드와 SDT 도구에 의해 자동 생성된 코드 및 CORBA 정합 및 블록간 통신 기능을 제공하기 위한 입출력 제어 및 SDT 커널 정합 기능을 제공하는 함수를 포함하여 실행 파일을 생성한다.

CORBA와 SDL 시스템간 연동 흐름은 (그림 7)과 같으며, 단계별 동작은 다음과 같다.

1) SDL 시스템을 실행시키면 시스템 초기화 및 CORBA 연동을 위한 객체 서비스 중개자를 초기화하고 CORBA 객체와 POA를 식별할 수 있는 정보 등을 포함하는 IOR (Interoperable Object Reference) 값을 생성하고, 클라이

언트로부터 요구를 기다린다.

2) SDL 시스템은 원격 시스템과의 시그널 기반 통신을 위한 입력 채널을 구성한다. CORBA 연동에서는 이용되지 않는다.
3) SDL 시스템은 CORBA 연동을 위하여 스켈레톤으로 응답 신호를 전달하기 위한 기능을 초기화한다.
4) 클라이언트로부터 요구가 들어오면, 원격 프로시쥬어 호출(또는 시그널)을 생성하고 연산의 매개인자를 할당한 후 SDL 시스템으로 전송하고, 응답이 필요한 경우 응답(reply)을 기다린다.
5) SDL 시스템은 원하는 연산을 실행하고, 원격 시그널이 있는 경우 출력 환경 함수를 호출하고, 시그널 대기 상태로 천이한다.
6) 스켈레톤에서는 외부 출력 환경 함수로부터 응답 시그널을 수신하면, 응답 정보를 마셜링해서 클라이언트에게 전달한다.



(그림 7) CORBA SDL 연동 흐름도

5.1 클라이언트 측 코드

클라이언트 프로그램은 IDL 컴파일러에 의해 생성된 스터브 코드와 SDL 시스템에 정의된 객체를 참조하기 위한 함수를 포함하며, 일반적인 CORBA에서의 객체 참조와 동일하다.

객체 참조를 위한 인터페이스 정의 파일 및 객체 참조를 위한 코드는 다음과 같다.

```
/* IDL 파일 */
interface Calculator {
        long multiply(in long first, in long second) with (priority
        = 2, timeout = 2);
};
/* 클라이언트 코드 */
orb = CORBA_ORB_init(&argc, argv, "uniorb-local-orb");
server = CORBA_ORB_string_to_object(orb, ior);
 ......
result = Calculator_multiply(server, first, second);
```

5.2 서버측 코드

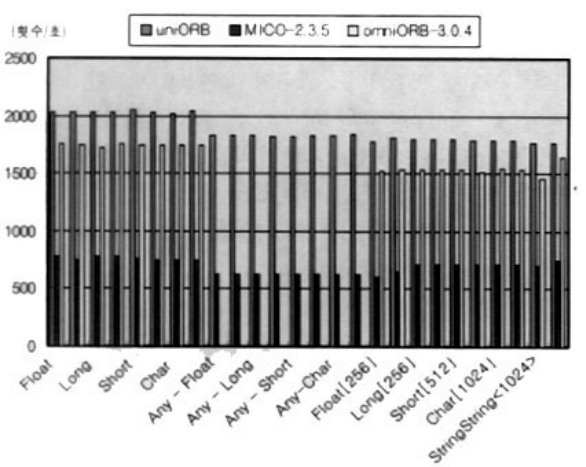
서버측 프로그램은 객체 서비스 중개자가 상호 연동 프로토콜(IIOP)을 통하여 클라이언트로부터 서비스 요구를 받으면, SDL 시스템 정합 기능이 포함된 IDL 컴파일러에 의해 스켈레톤 코드는 해당 객체에 대한 시그널 데이터에 매개 변수 값을 각각 매핑하고 SDT 커널을 통하여 시그널을 전달하고, 양방향 서비스인 경우 SDT 시스템의 출력 환경 함수로부터 응답 데이터 수신을 확인하는 신호를 대기한다.

```
& (((yPDP_senddata_in) (yOutputSignal)) → Param 2.c)))
= ( * ((uint32 *)_UNIORB_curptr));
/* Send to SDL System */
SDL_Output (yOutputSignal, (xIdNode *) 0);
/* Waiting for reply *.
status = pthread_cond_wait (& senddata_wait, &senddata_wait.mutex);
/* Return to Client */
```

6. 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, 256MB바이트의 주 기억장치의 시스템에 uniORB와 독일에서 공개용으로 개발된 MICO(Version 2.3.5) 및 루슨트테크놀러지에서 개발된 omniORB(3.0.4) 3가지 미들웨어를 대상으로 비교하였다.

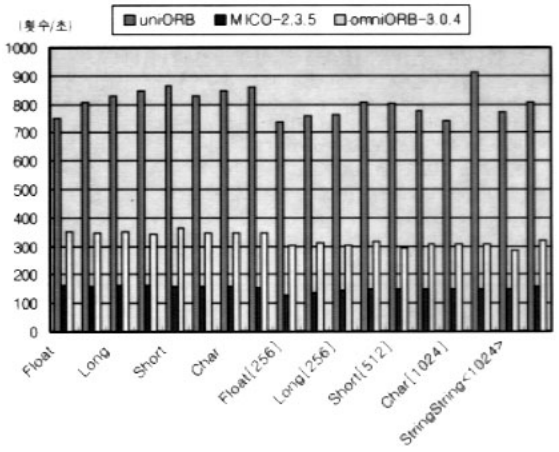
단일 클라이언트에서 동일한 함수를 1000회 호출하는 시험에서 (그림 8)에서 보는 바와 같이 본 논문에서 개발된 uniORB는 MICO에 비하여 약 2.5배, omniORB에 비하여 약 20%의 우수한 성능을 보였다.



(그림 8) 단일 클라이언트에서의 성능 비교

또한, 5개의 다중 클라이언트를 이용하여 성능을 측정한 결

과 (그림 9)와 같이 uniORB는 MICO에 비하여 각각의 데이터 타입에 대하여 약 3.5배, omniORB에 비하여 2.5배 이상으로 TCP/IP 기반의 IIOP 프로토콜을 최적화하여 구현함으로써 다중의 클라이언트의 요구 처리에서 상대적으로 낮은 성능 저하를 확인할 수 있었다.



(그림 9) 다중 클라이언트에서의 성능 비교

7. 결 론

본 논문에서는 개방형 통신시스템 개발에서 제어 계층과 응용 계층간 및 제어 계층 내부에서 기능 처리 블록간 시그널 기반의 분산 처리에서 미들웨어를 이용하여 함수 호출로 처리함으로써 시스템 개발의 효율성과 제어 시스템의 물리적 독립성을 제공하기 위하여 상용 운영체제에서 제한된 실시간 처리 기능을 제공하는 멀티쓰레드 기반의 미들웨어를 설계하고 구현하였다. 또한 통신시스템 개발툴인 SDT와 미들웨어를 결합할 수 있는 기능을 구현하여 시그널 기반의 응용에서 제공되는 객체를 함수 형태로 호출할 수 있도록 하였다. 그리고, 다양한 데이터 타입의 함수에 대한 성능을 분석한 결과 C++ 언어로 개발된 미들웨어에 비하여 C 언어로 개발된 uniORB는 약 20%에서 200%까지의 우수한 성능을 보였다.

향후 uniORB는 통신시스템용 미들웨어로 활용하기 위하여 이벤트 서비스, 통지 서비스 등의 다양한 서비스 개발, 성능 향상 및 시스템의 안정화 및 상용 미들웨어와의 호환성 관련 연구를 진행하여야 한다.

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 Submis-sion*, OMG Document orbos/99-02-12 ed., March, 1999.

[3] D. C. 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] The CORBA Integration, SDT manual, 1997.

[5] Using SDL to develop CORBA object implementations, Morgan Bjkander, Telelogic AB, 1997.

[6] Multiservice Switching Forum, http : //www.msforum.org.

[7] MSF, "Multiservice Switching Forum System Architecture Implementation Agreement 1.0," MSF00-044, Feb. 2000.

[8] K. H. K. Kim, "Object Structures for Real-Time Systems and Simulators," *Objects, Volume* 2. New York, NY : Wiley & Sons, 2000. *IEEE Computer*, p.6270, Aug. 1997.

[9] 이정규, 이순석, 김영부, 전경표, "개방형 멀티서비스 통합 교환 기술", ETRI 기술문서, 2000.

장 종 현

e-mail : jangjh@etri.re.kr
1988년 경북대학교 전자공학과(공학사)
2000년 충남대학교 전자공학과(공학석사)
1988년~1994 대우통신㈜ 종합연구소
현재 한국외국어대학교 전자공학과 박사과정
1994년~현재 한국전자통신연구원 교환전송기술연구소 선임연구원
관심분야 : 네트워크 보안, 이동통신, 미들웨어 등

이 동 길

e-mail : dgle@etri.re.kr
1983년 경북대학교 전자공학과(공학사)
1985년 한국과학기술원 전산학석사
1994년 한국과학기술원 전산학박사
1985년~현재 한국전자통신연구원 책임연구원
관심분야 : 컴파일러 구성론, 프로그램밍 언어론, 미들웨어 등

한 치 문

e-mail : chman@hufs.ac.kr
1977년 경북대학교 전자공학과(공학사)
1983년 연세대학교 대학원 전자공학과 (공학석사)
1990년 The University of Tokyo(동경대) 전자정보공학과(공학박사)
1977년~1983년 한국과학기술연구원(KIST) 연구원
1983년~1997년 한국전자통신연구원(ETRI) 선연, 책연 교환기술연구단 계통연구부장 역임
1997년~현재 한국외국어대학교 전자공학과 교수
관심분야 : ATM 통신망 및 교환, IMT-2000 네트워크, 네트워크 보안 등