논문 2004-41TC-2-3

개방형통신시스템을 위한 고성능, 고신뢰성 CORBA 플랫폼에 관한 연구

(A Study of High Performance and Reliable CORBA Platform for Open Communication Systems)

장 종 현*, 이 동 길*, 한 치 문**

(JongHyun Jang, DongGill Lee, and ChiMoon Han)

요 약

최근 분산컴퓨팅 시스템은 서로 이질적인 시스템간 상호 연동성 문제를 해결하기 위한 새로운 시스템 아키텍처를 제시하고 있다. 통신시스템과 같이 제한적 실시간, 고성능 및 신뢰성을 요구하는 CORBA는 유연성, 효율성 및 예측성 등이 지원되어야 한다.

본 논문에서는 개방형통신시스템용 소프트웨어를 지원하는 효율적인 CORBA 플랫폼 개발을 위해 2가지 방법을 제시한다. 첫째, 동일 호스트상에서 메시지 전달 오버헤드를 최적화하기 위해 공유 메모리 기반의 연동 프로토콜을 제시한다. 둘째, CORBA 플랫폼의 신뢰성 향상을 위해, 구현 객체 데이터베이스를 이용한 네이밍 서비스 서버 구현 방법과 제한적 실시간 특성 제공 방법을 제시한다.

따라서, 제안된 CORBA 플랫폼은 제한적 실시간 특성을 제공하고, 고성능이며 신뢰성 있는 네이밍 서버를 이용한 개방형 통신시스템에 적합한 소프트웨어 플랫폼임을 보인다. 본 논문은 플랫폼의 견고성이 요구되는 개방형통신시스템용 고성능 CORBA 플랫폼 설계 및 구현에 목적을 둔다.

Abstract

In this paper, the beam steering dipole phased array antenna systems for IMT-2000 base station have been designed. The designed beam steering dipole phased array antenna systems are constituted by the antenna part and the beam steering control system part. The antenna part is designed by the proposed flat dipole for the broadband characteristics, and the 8×8 dipole array antenna is constructed by the proposed flat dipole for the directional radiation pattern. Besides the vertical power divider is designed for the vertical power distribution. The beam steering control system part is designed the horizontal power divider for the horizontal power distribution, the 4-bit phase shifters and the driving circuit of phase shifters for the horizontal beam tilting. In order to evaluate a performance of the designed antenna systems, they were fabricated and the radiation characteristics were measured. From the measured results, we found that the horizontal beams were tilted by the each control signals, and the measured radiation characteristics showed good agreement with the design goals.

Keywords: CORBA, Middleware, Open Communication System

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.

CORBA는 하드웨어와 소프트웨어에 독립적인 객

* 정회원, 한국전자통신연구원(ETRI)

** 정회원, 한국외국어대학교 전자공학과 교수

(Dept. of Electronics Engineering Dankook University)

접수일자: 2003년8월7일, 수정완료일: 2004년2월6일

체지향 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(Remote Procedure Call) 등이 있다. 또한 CORBA를 이용한 분산 플랫폼은 일반적인 응용에서부터 실시간 통신 시스템 및 내장형 시스템으로 확장되어 사용되고 있다.

오늘날 급속한 기술 개발로 하드웨어는 점점 소형화, 고속화, 저가격화가 되는 반면에, 소프트웨어는 점점 대형화되어 속도도 느려지고 가격도 비싸져 분산시스템의 구축이 어려운 실정이다. 또한, 대규모 통신시스템, 항공 통제와 같은 특수 목적의 분산 어플리케이션들은 실시간 QoS의 보장이 요구된다. 그러나 기존의 CORBA 제품들은 효율적인 실시간 특성 및 QoS 요구 사항을 지원하지 못하므로 다음과 같은 조건의 CORBA의 개발이 요구된다[2].

- 다중 QoS 및 동시 제어를 위해 실시간 내장형 미들웨어 기술을 이용한 확장성 및 최적의 성능 만족
- 일반적 기술 기반의 고객화 된 구성을 제공할 수 있도록 미들웨어 기술을 통한 표준화

통신시스템이 표준 인터페이스를 기반으로 개방형 구조로 진화함에 따라 소프트웨어 개발 기간의 단축이 시스템의 경쟁력 확보의 관건이 되고 있다. 따라서, CORBA 기반의 미들웨어 기술을 통신시스템용 소프트웨어에 접목함으로써 개발 기간의 단축 및 재사용성이 향상되어 시스템의 구성의 유연성과 유지 관리에 효율성을 제고할 수 있다.

따라서, 본 논문에서는 네이밍 서버에서 구현 객체에 대한 데이터베이스를 이용한 신뢰성 제공과 동일 호스트상에서 실행되는 응용 블록간 분산 처리를 위한 공유 메모리 기반 연동 프로토콜을 이용한 개방형통신시스템용 고성능 CORBA 플랫폼 구현에 목적이 있다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 관련 분야의 규격 및 발전 방향을 분석하고, Ⅲ장에서는 개방형통신시스템을 위한 고성능, 고신뢰성 CORBA의 설계 및 구현에 대하여 기술하고, Ⅳ장에서는 플랫폼의 성능 분석을 통한 문제점 도출하고, Ⅴ장에서는 구현된 플랫폼의 적용 사례를 기술하고, Ⅵ장에서 결론을 맺는다.

Ⅱ. 관련 연구 분석

OMG에서 정의한 실시간 CORBA에 대한 주요 기능과 특징에 대한 연구를 통한 고성능, 고신뢰성 개방형통신시스템용 CORBA를 설계하고자 한다.

일반적으로 CORBA는 구현이 아닌 인터페이스를 정의하는 규격으로, 주요 특징으로는 객체 위치 투명성, 연결 및 메모리 관리, 매개 변수 (디)마셜링, 이벤트 및 요구의 (역)다중화, 오류 처리 및 고장 감내, 객체/서버의 활성화, 병행성 및 보안 기능을 제공한다. 현재 CORBA 인터페이스를 정의하는 규격에 800개 이상의 산업체가 참여하고 있다. CORBA는 언어, 운영체제, 하드웨어 및 네트워크 프로토콜의 이질적인 의존성으로부터 독립적인 응용을 구현할 수 있는 인터페이스를 제공한다.[1]

개방형 통신시스템에 분산된 객체는 범용 CORBA에서 제공되는 특성과 타임아웃, 시그널 전달 기능 등의 제한적인 실시간 서비스를 지원할 수 있는 특성을 제공하여야 한다.[2] 또한, 객체간 연결 채널은 TCP 또는 ATM PVC와 같은 연결형 서비스를 활용함으로써 채널 설정 시간을 최소화 한다. 범용 CORBA가 유닉스 운영체제를 기반으로 하는 시스템에서 제한적인 실시간 특성을 요구하는 통신시스템에 적용되기 어려운 점으로는 표준화된 실시간 응용을 지원하는 API가 존재하지 않으며 과도한 데이터 복사와 같은 지연 부하 때문에 성능을 만족하지 못한다.[3,5]

따라서, CORBA의 성능을 향상시킬 수 있는 다양한 연구가 진행되고 있는데 기존 연구에 대한 분석을 통하여 성능 향상 및 부하를 최소화할 수 있는 방법을 찾고자 한다. 객체간 호출을 지원하기 위하여 공통 데이터 표현(Common Data Representation)

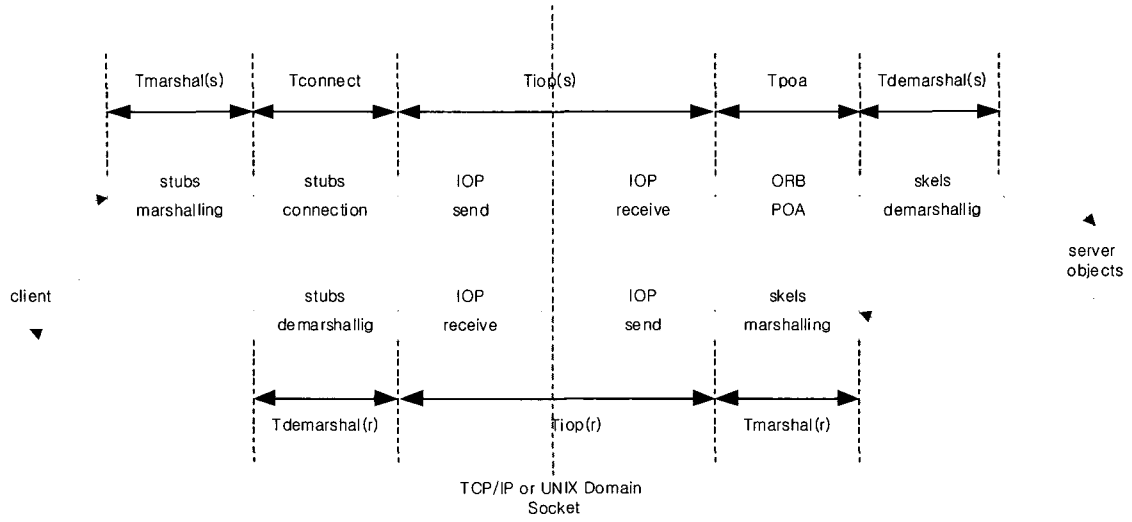

그림 1. CORBA 통신 모델
Fig. 1. CORBA communication model.

형식을 정의하고, 이 형식을 TCP/IP와 같은 전송 프로토콜을 통하여 전달하게 된다. 이때 전송 프로토콜에 따른 부가적인 메시지의 구성 요소는 다음과 같다.

- 프레임당 26바이트의 이더넷 헤더
- 패킷당 12 바이트의 IP헤더
- 패패킷당 24바이트(+ options)의 TCP 헤더

여기에 40 ~ 80 바이트의 GIOP 및 IIOP 메시지 헤더가 추가되므로 작은 크기의 메시지보다는 일정 크기 이상의 메시지 전송이 CORBA가 효율적이라 할 수 있다.

추가적인 오버헤더로는 객체중개자 내부에서 메시지를 처리하는데 소요되는 시간이 존재한다. 이 오버헤더는 두 가지 타입으로 구분할 수 있는데, 첫째는 (디)멀티플렉싱, 문맥 교환, 시스템 콜 실행 및 객체 활성화에 소요되는 고정된 오버헤더가 있고, 다양한 메시지와 매개 변수의 타입에 따라 (디)마셜링에 소요되는 가변 오버헤더가 있다. 이러한 오버헤더를 줄이기 위하여 *ORBexpress* 제품은 전체 전송시간을 5 ~ 20%의 성능 향상 효과를 확인할 수 있었다[4].

Ⅲ. 개방형통신시스템을 위한 고성능, 고신뢰성 CORBA 설계 및 구현: uniORB

통신시스템의 특성상 분산 하드웨어 장치에 기능이 내장되어 실행되는 특성을 지원하기 위한 CORBA는 실시간 처리, 고신뢰성, 고성능의 특성화된 기능을 요구한다. 일반적인 CORBA 응용의 경우, 물리적으로 분산된 응용에서는 소켓 기반의 연동 프로토콜을 적용하여야 한다.

본 논문에서는 동일 시스템 내에서 제한적 실시간 처리, 공유 메모리 기반의 연동 프로토콜과 네이밍 서버의 재기동시 이전 상태의 객체에 대한 정보를 기반으로 서비스의 연속성을 제공할 수 있는 고신뢰성 네이밍 서버의 설계 방법을 제안한다. 제안한 방법을 적용하여 통신시스템에 적용 가능한 고성능, 고신뢰성 CORBA 플랫폼(이하 uniORB라 한다)을 개발하고자 한다. 그리고, 본 논문에 적용된 CORBA는 미국 redhat사와 지원하에 진행되는 GNOME 프로젝트의 하나인 ORBit CORBA를 기본 플랫폼으로 사용하였다.[9]

3.1 공유 메모리 기반 인터넷 연동 프로토콜 설계

IIOP(Internet Inter-ORB Protocol)은 TCP/IP 기반의 네트워크 상에서 여러 개의 객체 중개자(ORB) 간의 통신을 위해 표준 메시지 형식과 공통 데이터 표현형의 세트로 정의된 GIOP(General Inter-ORB Protocol) 메시지를 교환하는 프로토콜을 의미한다. 일반적인 CORBA 통신 모델을 살펴보면 그림-1과 같이 두 컴포넌트간 통신을 위하여 TCP/IP 또는 유

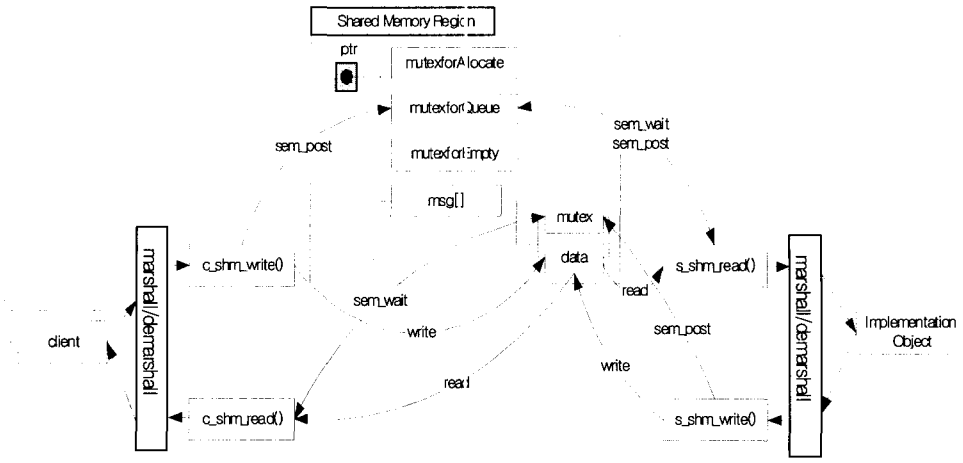

그림 2. 공유메모리 기반 IIOP 모델
Fig. 2. Shared memory based IIOP model

닉스 도메인 소켓을 주로 이용한다.

개방형통신시스템용 고성능 플랫폼 설계를 위하여 썬 마이크로시스템즈사의 솔라리스2.7 운영체제와 UltraSparc 167MHz, 192MB바이트의 주 기억장치를 가진 시험환경에서 기본 CORBA 플랫폼의 성능 분석을 통하여 최적화할 수 있는 요소를 찾고자 한다.

따라서, 클라이언트에서 서버로의 CORBA 서비스 요청이 항상 일정한 패킷 전달능력을 유지한다고 가정할 때, CORBA의 성능은 다음과 같은 속성에 의하여 결정되게 된다.

본 논문에서는 3개의 *in, out, inout* 매개변수와 동일한 타입의 값을 반환하는 서비스에 대한 성능 분석을 통하여 기존 모델에서 성능을 향상시킬 수 있는 기준을 도출하고자 한다.

다양한 데이터 타입에 대하여 성능은 표-1과 같은 결과를 얻을 수 있었다.

표 1. CORBA 구성요소 별 실행시간 비교 (단위:초)
Table. 1. Comparison of execution time by CORBA constituent(Unit: sec.)

구분	Ttrans	Tpoa	Tmarshal	Ttotal	Ratio
Character	0.007552	0.000067	0.000207	0.007826	94 %
Long	0.003467	0.000078	0.000201	0.003747	93 %
Float	0.003776	0.000066	0.000203	0.004045	88 %
Double	0.005549	0.000067	0.000207	0.005823	91 %
String	0.005071	0.000063	0.000249	0.005384	90 %
Sequence	0.003208	0.000064	0.000264	0.003536	84 %

여기서, 마셜링 시간 *Tmarshal*는

$$Tmarshal = \Sigma(Tmarshal(s) + Tdemarshal(s) + Tdemarshal(r) + Tdemarshal(r)) \quad (1)$$

서버측의 객체중개자 사용하는 시간 *Tpoa*는 요청된 서비스를 해석하고, 구현 객체 저장소에서 해당 객체를 찾아서 서비스를 기동시키는데 소요되는 시간을 말한다.

전달 계층을 통하여 클라이언트 측에서 마셜링된 메시지를 서버측으로 전달하고, 결과값을 반환하는데 소요되는 시간 *Ttrans*는

$$Trans = \Sigma(Tiop(s) + Tiop(r) + Tconnect) \quad (2)$$

따라서, 하나의 CORBA 서비스를 처리하는데 소요되는 시간 *Ttotal*은

$$Ttotal = \Sigma(Tmarshal + Tpoa + Ttrans) \quad (3)$$

수식 (2)와 (3)으로부터 연동프로토콜의 전달에 소요되는 비율 *Ratio*는

$$Ratio = (Ttrans/Ttotal)*100$$

메시지를 연동 프로토콜을 통하여 전달하는데 소요되는 *Ttrans* 시간이 차지하는 비율은 데이터 타입에 따라 약간의 차이가 있으며, 또한 전달되는 매개 변수의 데이터 크기에 따라 변화는 있겠지만 약 80% 이상의 시간이 소요됨을 확인할 수 있었다.[4]

다음은 동일시스템내의 클라이언트/서버 모델에서 프로세스간 통신 방법으로 공유메모리와 TCP/IP를 사용하여 다른 크기의 데이터에 대하여 1000번씩 전달하는데 소요되는 시간을 분석하였다.

표 2. 통신 모델에 대한 수행시간 비교 (단위 :초)
Table. 2. Comparison of performance time by communication model(Unit: sec.)

구분	1KB	4KB	16KB	32KB	64KB
SHM	0.075521	0.078961	0.078405	0.077042	0.076789
TCP	0.083218	0.086157	0.164015	0.305352	0.610594

표-2에서 보듯이 시스템 내부에서 공유 메모리를 이용하는 경우 데이터 크기에 상관없이 거의 동일한 전달 능력을 보이는 반면, TCP/IP 프로토콜에서는 데이터 크기에 민감하게 실행시간이 급격히 증가하였다.

따라서, 본 논문에서는 위의 두 실험의 분석을 통하여 통신시스템과 같이 실시간, 고성능 CORBA 플랫폼의 요구사항을 만족하고, 성능을 최적화하기 위하여 시스템 내부 객체간 통신을 위한 전송 오버헤드를 최소화할 수 있는 공유 메모리 기반의 연동 프로토콜을 제안하고자 한다.

공유 메모리 기반의 인터넷 연동 프로토콜의 동작 모델은 그림-2와 같으며, 아래 그림-3의 알고리즘과 같이 동작하도록 설계하였다.

1) 서버측 연동 프로토콜에서 공유 메모리 영역을 할당하고 필요한 자료 구조에 대해 초기화 작업을 수행한 후, 클라이언트로부터 메시지 수신을 기다린다.

2) 클라이언트는 통신을 위해 고유의 채널을 할당하고, 주어진 공유 메모리상의 데이터 영역에 메시지를 복사한 후 세마포어 카운터를 증가시킨다.

3) 서버측 연동 프로토콜은 세마포어 값을 검사하고, 공유 데이터 영역으로부터 해당 메시지를 읽어 객체중개자로 전달하고, 세마포어 카운터를 증가시킨다.

4) 서버측 연동 프로토콜은 응답이 필요한 경우 응답 메시지를 공유 데이터 영역에 복사하고 세마포어 값을 증가시킨다.

5) 클라이언트측 연동 프로토콜은 자신의 데이터에 대한 세마포어가 증가되면, 해당 메시지를 읽어 객체중개자로 전달한다.

```
// Client Side IIOP
Declaration variables
SharedMemoryBuffer {
    Lock MUTEX_LOCK[M];
    Channel Integer[N];
    msg[SIZE] GIOPMessage;
    keelAlicve Integer;
    reply_expected Integer;
};
SharedPtr *Ptr;
Initialization
        Ptr = MemoryMap(NULL, fd, flags);
Action
1: Make Client Shared Memory Buffer Create
2: Channel Allocation for Client Process
3:
4: Construct GIOPMessage
5: Semaphore TryWait(Lock)
6: Write GIOPMessage to SharedBuffer
7: Semaphore post(Lock)
8: Semaphore wait(Lock)
9: if oneway is not set then
10:         Wait for ReplyMessage from Server
11:         Handle ReplyMessage
12: fi
13: Channel Close
```

```
// Server Side IIOP
Declaration variables
SharedMemoryBuffer {
    Locks MUTEX_LOCK[M];
    msg[SIZE] GIOPMessage;
};
SharedPtr *Ptr;

Initialization
        Ptr = MemoryMap(NULL, fd, flags);
        Semaphore Initialize (Lock[M])
Action
1: Make Server Shared Memory Buffer Create
2: While loop
3: do
4:          Semaphore Trywait(Lock)
5:          Wait for Service Request from Client
6:          Read GIOPMessage from SharedBuffer
7:          Semaphore Post(Lock)
8:          Decode GIOPMessage
9:          Handle Incoming Message
10:         if Reply_required then
11:                 Send Reply GIOPMessage to
    Client
12:             Semaphore Post(Lock)
13:         fi
14: done
```

그림 3. 제안한 모델의 동작 알고리즘
Fig. 3. Algorithm of proposed model.

공유 메모리 기반 연동 프로토콜 구현에서 발생할 수 있는 다음과 같은 문제점을 확인할 수 있었다. 첫째, 서버측 연동 프로토콜이 공유 메모리 영역을 생성할 때까지 클라이언트의 공유 메모리 접근을 방지하여야 한다. 둘째, 서버측 연동 프로토콜이 응답을 반환할 때까지 새로운 서비스를 시작하지 않아야 한다. 셋째, 객체중개자가 구현 객체를 수행하고 있는 도중에 실행중지가 발생하게 되면, 공유 메모리에 구현 객체의 비활성화 정보를 전달하여 세마포어의 오류를 방지하여야 한다.

이와 같은 문제점을 해결하기 위하여 서버측 연동 프로토콜은 구현 객체가 실행되는 동안 클라이언트 프로그램이 사용자에 의한 실행 중지 및 프로그램 오류에 의해 서비스 중지가 발생되는 경우 클라이언트측에서 제공하는 *keepalive* 값을 검사하여 세마포어 정보를 초기화하는 기법으로 문제점을 해결하였다.

3.2 고신뢰성 네이밍 서비스 설계

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 연결짓는 것을 바인딩이라 하는데, 이름은 객체의 이름을 나타내는 문자열의 식별자와 이름을 설명하기 위한 종류(kind)라는 두 개의 속성을 가지는 구조체이다.

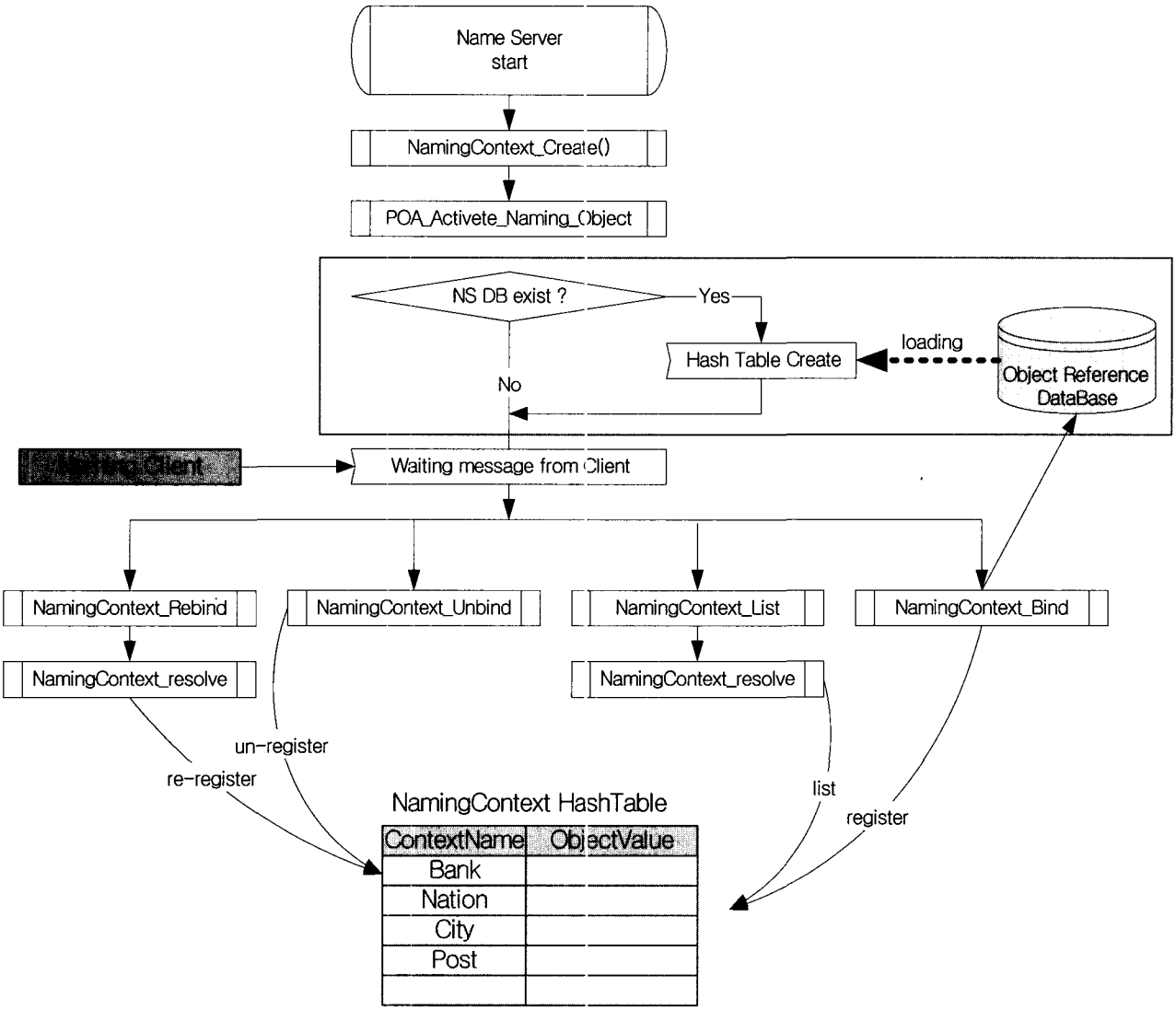

그림 4. 객체 DB를 이용한 네이밍 서버 기능 흐름도
Fig. 4. Function flow of naming server using object database.

어떤 객체를 바인딩하면 바인딩 세트를 저장하는 컨테이너 객체인 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[1].

다중의 클라이언트가 여러 개의 제어 보드에 의해 동작하는 구현 객체로 접근할 경우 네이밍 서비스를 이용한다. 일반적인 CORBA 기반 응용에서 하드웨어 오류 또는 다른 원인에 의해 네이밍 서버가 다운되었다가, 복원되는 경우 네이밍 서버 및 구현 객체를 재실행하므로 서비스 중단이 발생하게 된다. 통신시스템의 특성인 무정지형 서비스를 제공하기 위하여 본 논문에서는 네이밍 서버만의 재실행으로 객체 참조 데이터베이스에 저장된 구현 객체에 대한 바인딩 정보를 이용하여 네이밍 그래프를 재구성하여, 시스템의 신뢰성을 향상시키는 방법을 제안한다.

제안된 네이밍 서버에서 구현 객체가 바인딩을 요청하면 네이밍 그래프를 생성하면서 그림-4와 같은 절차로 재실행시 기존의 객체 서비스를 제공하기 다음과 같은 네이밍 객체 데이터베이스를 생성한다.

```
/* uniORB Naming Service DataBase File */
1                /* Number of Objects */
========================================
1                /* 1st Implementation Object */
Bank             /* Object Name */
IOR:000000000000002b49444c3a6f6d672e6f72672f436f7
34e616d696e672f4e616d696e67436f6e746578744578743a31
2e300000000000100000000000000380001000000000000f313
2392e3235342e3137312e35370000878f000000000018000000
0000f9f7fcfd5afc030000000100fdf5fff1a5d7fc
```

여기서 네이밍 서버를 재실행하면 네이밍 정보 데이터베이스의 존재를 확인한 후, 네이밍 객체 데이터베이스에 존재하면 *CORBA_string_to_object*() 함수를 이용하여 구현 객체 정보를 생성하고 *CORBA_NamingContext_bind*() 함수를 이용하여 바인딩을 요청한다.

3.3 제한적 실시간 기능 설계

유닉스 기반의 범용 운영체제 환경에서 실시간 운영체제가 제공하는 실시간 특성을 제공하기는 쉽지 않다. 따라서, 본 논문에서는 실시간 CORBA 규격에서 시간 제어 부분으로 서버로부터 무응답을 방지 하기 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 시간동안 응답이 없을 경우 오류 타입을 CORBA_TIMEOUT_SERVER로 정의하고, 할당된 입출력 버퍼를 해제하고 반환 값을 0으로 정의하여 제어를 객체중개자로 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 접근 할 수 있도록 하는 기법으로 주로 멀티쓰레드 기반 응용 프로그램의 콜백 함수를 활용한다.

```
long atTimeReq(in long input) with
```

(at=200212091955);/ 2002년 12월 9일 19:55 */*
long atTimeReq(in long input) with(at=+3600);
/ 현재 시간으로부터 1시간 후 */*

Ⅳ. CORBA 성능 분석

성능 분석은 널(null) 함수로 작성된 서버 프로그램에 대하여 OMG 규격에서 정의한 데이터 타입을 이용하였다. 시스템 구성은 서버와 4개의 클라이언트로 구성하였으며, 4개의 클라이언트에서 동시에 서비스를 요청한 후, 결과를 반환 받는 형태로 CORBA 처리율을 비교하였다. 성능 분석 시스템 환경은 썬 마이크로시스템즈사의 솔라리스 버전 2.7 운영체제와 UltraSparc 167MHz, 192MB바이트의 주 기억장치의 시스템을 사용하였다.

기존 범용 및 상용 CORBA 플랫폼은 공유 메모리 기반의 IIOP 프로토콜을 지원하지 않아 비교를 할 수 없으므로, 평가 대상을 동일 플랫폼의 IIOP 프로토콜간 성능으로 제한하였다.

먼저, 플랫폼의 신뢰성을 위하여 독일에서 공개용으로 개발된 MICO(Version2.3.5) 및 루슨트테크놀러지에서 개발된 omniORB(3.0.4) CORBA를 대상으로 비교하였다. [8,9]

5개의 다중 클라이언트에서 다양한 데이터 타입의 함수[10]를 1000회 호출하여 소요되는 시간을 비교한 결과 그림-5와 같이 본 논문에서 제안한 CORBA 플랫폼인 uniORB는 C++언어로 구현된 MICO에 비하여 각각의 데이터 타입에 대하여 약 2.5배, omniORB에 비하여 15% 의 성능 향상을 보였다.

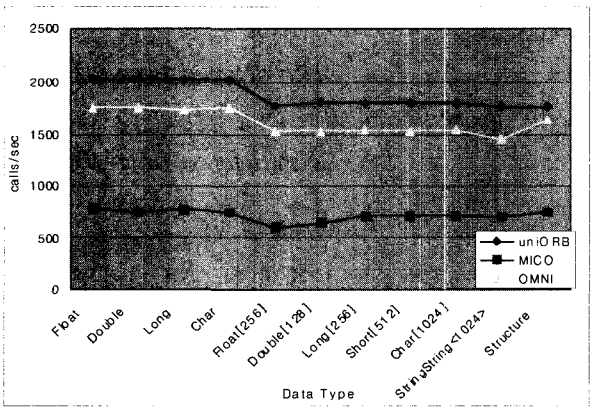

그림 5. 다중 클라이언트에서의 성능 비교
Fig. 5. Comparison of performance in multi-clients.

다음은 본 논문에서 제안한 방식의 공유 메모리 기반의 IIOP 모델과 기존의 TCP/IP 및 유닉스 도메인 소켓 기반 플랫폼에 대하여 성능을 측정하였다.

첫 번째 성능 측정은 클라이언트에서 다양한 형태의 데이터 타입의 서비스를 1000회 요청하는 시험을 수행하였다. 데이터 유형에 따른 처리율을 그림-6에 나타냈다. 그림에서 보면, 공유 메모리 기반의 연동 프로토콜을 이용하면 경우 유닉스 도메인 소켓에 비하여 약 50%, TCP/IP 기반의 인터넷 연동 프로토콜에 비하여 2배의 처리율이 향상됨을 알 수 있다.

이와 같은 성능 향상은 데이터 타입에 따른 마셜링/디마셜링 및 객체중개자에서 해당 객체를 찾고 활성화에 소요되는 시간은 동일하지만 TCP/IP 기반 모델의 커널 내부에서의 데이터 복제 시간에 따른 오버헤더가 발생하지 않으므로 2배 이상의 성능 향상을 보였다.

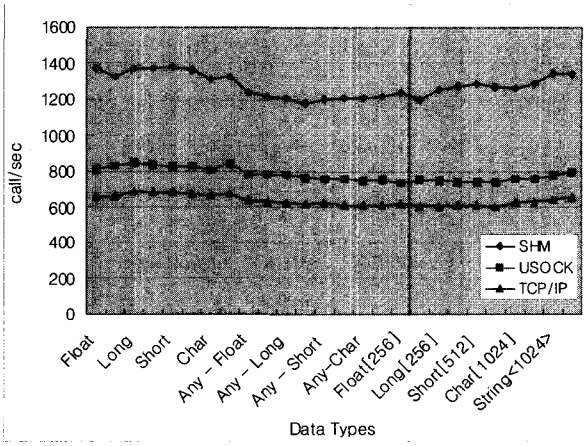

그림 6. 데이터 타입에 따른 처리율 비교
Fig. 6. Comparison of throughput by data type.

두 번째 성능 분석은 전달되는 문자 타입의 데이터 크기를 변화시키고, 서버측 구현 코드에서 전달된 데이터 크기를 비교한 후 결과값을 반환하는 실험을 수행하여 처리율을 비교하였다. 그림-7는 1000회 서비스 요청/응답에 대해 소요되는 시간을 나타냈다.

그림에서 보면, 공유 메모리 기반의 연동 프로토콜이 데이터 크기가 작으면 성능 향상은 상당히 개선되지만, 데이터 크기가 증가하면 다른 방식에 비

해 성능 차이는 크게 개선되지 않는다. 이러한 이유는 데이터 복사에 요구되는 시간이 상대적으로 CORBA의 데이터 형식 표현을 변경하는 마셜링/디마셜링 및 객체중개자의 처리 시간에 비하여 많기 때문이다.

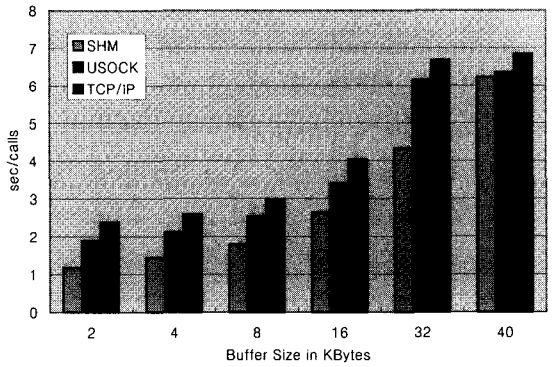

그림 7. 데이터 크기에 따른 처리율 비교
Fig. 7. Comparison of throughput by data size.

본 논문의 성능 분석 결과를 통해, 공유 메모리 기반의 연동 프로토콜은 시스템의 주 기억장치 메모리가 충분하고, 가능한 작은 크기의 데이터를 빈번하게 처리하는 응용에 적합하다. 또한, 수십 개의 클라이언트가 동시에 하나의 서버에 접근하는 응용에 적용하기 위하여 보다 향상된 공유 메모리 처리 기법이 요구된다.

제한적 실시간 기능에 대한 평가는 적용 사례의 라우터 운용 보전시스템의 자원 모니터링 기능 처리에서 라인 프로세서로부터 일정 시간 동안 무응답과 주기적인 통계 정보 수집 기능 처리에 적용하여 구현하였으며, 신뢰성 기능은 OXC 운용 관리 플랫폼을 통하여 확인할 수 있었다.

V. 적용 사례

OXC(Optical Cross Connector) 관리시스템은 그림-6과 같이 OXC 운용관리를 위한 분산관리 플랫폼으로 개발되었으며, 사용자 정합장치와 운용 관리 시스템간은 상용 CORBA 제품인 VisiBroker와 uniORB의 연동을 통하여 객체를 액세스하고 있음을 보여준다.[6] 각각의 제어 보드들은 객체지향 모델링 과정에서 만들어진 OXC 운용관리객체를 수행할 수 있도록 설계하였으며, OXC 운용 관리 시스템은 시스템 운영자를 위한 GUI 환경을 제공한다[6]. 본 응용에서는 여러 개의 보드로 구성된 서버 제어 장치에 위치한 객체 서비스에 본 논문에서 구현된 네이밍 서비스를 적용함으로써 장치의 탈/부착에 의한 이중화에서 서버 제어장치에 실장된 프로그램의 재실행 없이 서비스를 계속할 수 있는 신뢰성을 제공하게 되었다.

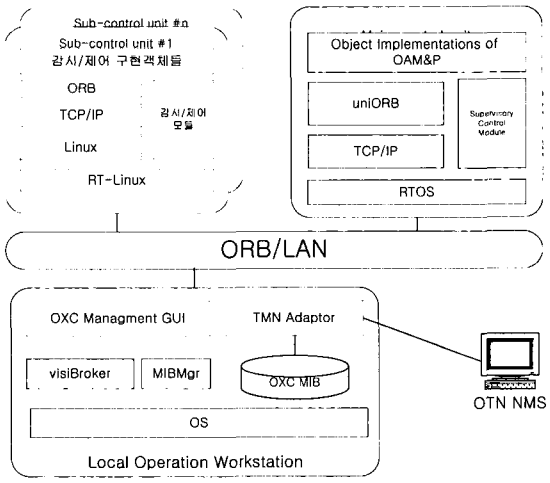

그림 8. OXC의 OA&M 관리 플랫폼 구조
Fig. 8. OXC operation management and management control platform structure.

VI. 결 론

본 논문에서 제시한 통신시스템을 위한 CORBA는 범용 CORBA 제품과는 달리 고성능을 요구하는 통신시스템용 소프트웨어 개발에 적용하기 위해 설계되었다. 기존 시그널 기반의 통신 기능을 블록간 CORBA의 표준 인터페이스 정의를 통하여 제공함으로써 소프트웨어 생산성과 재활용성을 향상시킬 수 있다. 특히 동일 호스트상에서 실행되는 객체에 대한 용이한 접근과 네이밍 데이터 베이스를 이용한 플랫폼의 신뢰성을 제고 및 제한적 실시간 처리 기능을 제공하는 고성능, 고신뢰성 CORBA 설계 및 구현이 가능함을 확실하게 하였다.

또한, 본 논문에서 제시한 방법이 다양한 데이터 타입의 함수에 대해 성능을 분석한 결과 기존의 TCP/IP 프로토콜 기반의 인터넷 연동 프로토콜에 비하여 약 2배 이상의 처리율을 나타내었다.

결론적으로, 제안된 CORBA 플랫폼을 기반으로 하드웨어 독립적이며, 확장성과 재사용성을 향상시킬 수 있는 소프트웨어의 구조의 표준화된 개방형 통신시스템의 설계가 가능하여 소프트웨어의 품질 향상과 시스템의 경쟁력을 확보할 수 있을 것으로 기대된다.

향후 uniORB가 내장형 시스템에서 실행되는 통신용 소프트웨어에 적용하기 위해서는 메모리 사용의 최적화, 데이터 표현 형식의 최적화를 통한 성능 향상과 이벤트 서비스, 통지 서비스 등의 다양한 서비스 개발 및 시스템의 안정화 관련 연구를 수행하여야 한다.

참 고 문 헌

[1] OMG, The Common Object Broker: Architecture and Specification, Revision 2.3, OMG Document Formal/98-12-01, June 1999.

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

[3] IMAI Yuji, SAEKI Toshiaki, CrispORB: High Performance CORBA for Systmem Area Network,, ISHIZAKI Tooru, KISHIMOTO Mitsuhiro software Laboratory

[4] Cathy Hrustich, CORBA For Real-Time, High Performance and Embedded Systems, Objective Interface Systems

[5] Aniruddha S. Gokhale and Douglas C. Schmidt, Optimizing a CORBA Internet Inter-ORB Protocol (IIOP) Engine for Minimal Footprint Embedded Multimedia Systems, IEEE Journal on Selected Areas in Communications, Vol. 17, No. 9, Sep. 1999

[6] 박수명, 이상화, 남현순, 주성순, CORBA 기반의 OXC 운용 관리 플랫폼 설계, Optoelectronics and Microelectronics 2001, Nov. 2001

[7] Puder and K. Rmer, MICO - User and Reference Manual. Technical Report TR-98-031, International Computer Science Institute, Berkeley.

[8] Sai-Lai Ro, David Riddoch and Duncan Grisby, The omniORB version 3.0 User's Guide, AT&T Laboratories Cambridge, May 2000

[9] GNOME/ORBit, http::/orbit-resouce.sourceforge.net

[10] Petr Tma, Adam Buble, Technical Report on Open CORBA Benchmarking, http://nenya.ms.mff.cuni.cz/~bench/package/Benchmarking-Techreport-200101.pdf

저 자 소 개

장 종 현(정회원)

1988년 경북대학교 전자공학과(공학사)

2000년 충남대학교 전산학과(공학석사)

2004년 한국외국어대학교 전자정보공학과(박사 졸업 예정)

1988년 2월~1994 2월 대우통신㈜ 종합연구소

1994년 3월~현재 한국전자통신연구원 광대역통합망연구단 선임연구원

<주관심분야: 네트워크 보안, 이동통신, 미들웨어 등>

이 동 길(정회원)

1983년 경북대학교 전자공학과(공학사)

1985년 한국과학기술원 전산학석사

1994년 한국과학기술원 전산학박사

1985년~현재 한국전자통신연구원 광대역통합망연구단 책임연구원.

<주관심분야: 컴파일러 구성론, 프로그램밍 언어론, 미들웨어 등>

한 치 문(정회원)

1977년 경북대학교 전자공학과(공학사)

1983년 연세대학교 대학원 전자공학과(공학석사)

1990년 The University of Tokyo(동경대)전자정보공학과(공학박사)

1977년2월~1983년3월 한국과학기술연구원(KIST) 연구원

1983년4월~1997년2월 한국전자통신연구원(ETRI) 선연, 책연 교환기술연구단 계통연구부장 역임

1997년3월~현재 한국외국어대학교 전자공학과 교수

<주관심분야 : ATM 통신망 및 교환, IMT-2000 네트워크, 네트워크 보안 등>