광역 객체 컴퓨팅 환경에서 부하를 고려한 통합 바인딩 서비스의 설계 및 구현

(Design and Implementation of Integrated Binding Service of Considering Loads in Wide-Area Object Computing Environments)

정 창 원 † 오 성 권 †† 주 수 종 †††
(Chang-Won Jeong) (Sung-Kwun Oh) (Su-Chong Joo)

요 약 최근 분산 컴퓨팅 환경은 급진적으로 광역화되고, 이질적이며, 연합형태의 광역 시스템 구조로 변화하고 있다. 이러한 환경은 네트워크상에 광범위한 서비스를 제공하는 통신 네트워크 기반에서 구현된 수많은 객체로 구성된다. 더욱, 지구상에 존재하는 모든 객체들은 이름이나 속성에 의해 중복된 특성을 갖는다. 그러나 기존의 네이밍이나 트레이딩 메커니즘은 독립적인 위치 투명성 결여로 중복된 객체들의 바인딩 서비스 지원이 불가능하다. 서로 다른 시스템 상에 존재하는 중복된 객체들이 동일한 서비스를 제공한다면, 각 시스템의 부하를 고려하여 클라이언트의 요청을 분산시킬 수 있다. 이러한 이유로 본 논문에서는 광역 컴퓨팅 환경에서 중복된 객체들의 위치 관리뿐만 아니라 시스템들간의 부하 균형화를 유지하기 위해서 최소부하를 갖는 시스템에 위치한 객체의 선정하여 동적 바인딩 서비스를 제공할 수 있는 새로운 모델을 설계하고 구현하였다. 이 모델은 네이밍 및 트레이딩 기능을 통합한 서비스에 의해 중복된 객체들에 대한 단일 객체 핸들을 얻는 부분과, 얻어진 객체 핸들을 사용하여 위치 서비스에 의해 하나 이상의 컨택 주소를 얻는 부분으로 구성하였다. 주어진 모델로부터, 우리는 네이밍/트레이딩 서비스와 위치 서비스에 의한 전체 바인딩 메커니즘의 처리과정을 나타내고, 통합 바인딩 서비스의 구성요소들에 대한 구조를 상세하게 기술하였다. 끝으로 구현 환경과 구성요소에 대한 수행 화면을 보였다.

키워드 : 광역 객체 컴퓨팅, 통합 바인딩 서비스, 네이밍 서비스, 트레이딩 서비스, 위치 서비스

Abstract In recent years, distributed computing environments have been radically changing to a structure of global, heterogeneous, federative and wide-area systems. This structure's environments consist of a lot of objects which are implemented on telecommunication network to provide a wide range of services. Furthermore, all of objects existing on the earth have the duplicated characteristics according to how to categorize their own names or properties. But, the existing naming or trading mechanism has not supported the binding services of duplicated objects, because of deficiency of independent location service. Also, if the duplicated objects which is existing on different nodes provide the same service, it is possible to distribute the client requests considering each system's load. For this reason, we designed and implemented a new model that can not only support the location management of replication objects, but also provide the dynamic binding service of objects located in a system with minimum overload for maintaining load balancing among nodes in wide-area object computing environments. Our model is functionally divided into two parts; one part is to obtain an unique object handle of replicated objects with same property as a naming and trading service, and

· 본 연구는 한국과학재단 목적기초연구(2000-1-30300-010-2) 지원으로 수행되었음.
† 비 회 원 : 원광대학교 컴퓨터공학과
mediblue@wonkwang.ac.kr
†† 비 회 원 : 원광대학교 전기전자및정보공학부 교수
ohsk@wonkwang.ac.kr
††† 종신회원 : 원광대학교 컴퓨터공학과 교수
scjoo@wonkwang.ac.kr
논문접수 : 2002년 8월 26일
심사완료 : 2003년 2월 4일

the other is to search one or more contact addresses by a location service using a given object handle. From a given model mentioned above, we present the procedures for the integrated binding mechanism in design phase, that is, Naming/Trading Service and Location Service. And then, we described in details the architecture of components for Integrated Binding Service implemented. Finally, we showed our implement environment and executing result of our model.

Key words : Wide-Area object computing, Integrated Binding Service, Naming Service, Trading Service, Location Service

1. 소 개

최근 분산 컴퓨팅의 환경은 인터넷 기술의 발전으로 인하여 광역의 분산 컴퓨팅 환경으로 변화되고 있다. 이러한 환경에서 분산 투명성[1]을 제공하기 위한 연구가 활발하게 진행되고 있다. 대표적인 연구로는 OMG의 CORBA, Microsoft DCOM 등이 있다. 이들 연구들은 공통적으로 분산 시스템을 구성하는 객체들의 효과적인 관리와 검색 서비스[2]를 제공하기 위한 디렉토리[3, 4], 이름 서비스[5] 그리고 트레이딩 서비스[6]를 제공한다. 이러한 서비스는 전통적으로 객체 대 주소 매핑(object-to-address)을 제공하고 있다. 이와 같은 방법은 객체의 위치가 변경하였을 때, 객체의 식별자와 위치 정보의 갱신을 피할 수 없게 한다. 그리고 객체의 이름을 사용자가 변경하였을 때에도 마찬가지로 갱신해야만 한다[7, 8].

또한 이러한 환경에서 동일한 서비스를 제공하는 수많은 객체들이 네트워크 상에 분산되어 존재하게 된다. 이로 인하여 클라이언트들이 임의의 검색 서비스를 이용하여 객체를 찾을 경우, 해당 검색 서비스에서 관리하는 객체들은 이름과 속성으로 중복되어 클라이언트의 요구사항에 대응하는 객체를 찾기란 어렵다. 여기에는 가장 최적의 객체를 선정하기 위한 전략이 필요하다.

따라서, 본 논문에서는 광역 컴퓨팅 환경에서 중복된 객체들의 위치 관리뿐만 아니라 중복된 객체들 중에 부하를 고려하여 최적 객체를 선정 할 수 있는 통합 바인딩 서비스 모델을 제시한다. 이 모델은 서비스를 제공하는 객체에 대해 서비스 제공자가 정의한 이름과 속성을 비스 오퍼 단위로 저장하고, 이를 근거로 클라이언트가 검색할 수 있는 네이밍/트레이딩 서비스와 객체들의 위치 정보를 컨택 레코드로 관리하는 위치 서비스로 구성하였다. 객체들의 이동을 지원하기 위해 네이밍/트레이딩 서비스와 위치 서비스를 각각 독립적으로 운영하며, 이들 서비스가 관리하는 정보의 매핑은 객체를 식별할 수 있는 객체 핸들에 의해 이루어진다. 이로 인하여 객체의 위치가 변경되더라도 위치 서비스가 관리하는 컨택 레코드의 위치 정보만 변경하도록 하였다. 또한, 중복된 객체들 중에 하나의 객체를 선정하기 위해 객체가 위치한 시스템의 부하정보를 이용하여, 부하가 가장 적은 객체를 선정하도록 하였다[9]. 광역 객체 컴퓨팅을 위한 하부 구조는 CORBA 사양을 따르는 상용화된 미들웨어로 Borland의 VisiBroker 4.1을 사용하였으며, 통합 바인딩 서비스의 구성요소인 네이밍/트레이딩 서비스와 위치 서비스는 자바(Java2 SDK 1.3.1)로 개발하였다. 서비스 오퍼와 컨택 레코드 정보를 관리하기 위해 SQL Server 2000을 이용하였으며, 각 시스템의 부하정보를 획득하기 위해 LSF 4.1을 이용하였다.

본 논문의 구성을 보면, 다음 2장에서는 기존의 네이밍 서비스와 트레이딩 서비스의 기본적인 원리에 대해 설명하고, 부하 균형화에 대해서 기술한다. 3장은 우리가 제안한 통합 바인딩 서비스 모델에 대해 기술하였다. 4장에서는 이에 대한 개발환경과 서비스 수행 결과화면을 보인다. 끝으로 5장에서는 결론 및 향후 연구 내용에 대해 다룬다.

2. 관련 연구

본 장에서는 네이밍 서비스와 트레이딩 서비스의 원리에 대해서 기술하고, 또한 부하 균형을 위한 전략에 대해 살펴본다.

2.1 네이밍 서비스

분산 객체를 참조하기 위해서는 객체의 유일한 이름인 객체 레퍼런스를 이용해야한다. 그러나 객체 레퍼런스는 시스템에 의존적인 형태로 사용자가 다루기 어렵다. 따라서, 이를 문자열 형태의 객체 이름을 실제 객체의 이름인 객체 레퍼런스로 매핑시켜 주는 서비스를 제공한다. 이를 네이밍 서비스라 한다. 네이밍 서비스는 다음 그림 1과 같이 디렉토리 구조의 네이밍 규칙을 지원하다. 가장 하위 레벨의 a2는 sys와 bin이라는 네이밍 컨텍스트(Naming Context) 아래 위치한다. 이를 sys::bin::a2라는 방식으로 구별한다. 네이밍 컨텍스트는 이름들의 범위를 정의한다. 네이밍 서비스에서 제공하는 주요 메소드는 다음과 같다.

- Bind : 객체 레퍼런스에 대한 문자열 이름을 제공한다.

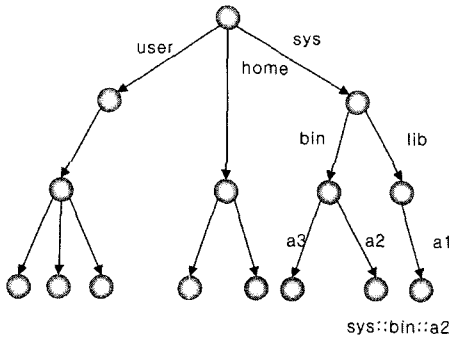


그림 1 네이밍 서비스의 원리

- Resolve : 주어진 문자열 이름에 해당하는 객체 레퍼런스를 제공한다.

다음 표 1은 기존의 다양한 네이밍 시스템의 특징을 보인다.

표 1 다양한 네이밍 시스템의 특징

	X.500	DNS	URLs	URNs	UNIX file	SSP chains
Location transparent	Yes	Yes	No	Yes	Yes/No	Yes
Human redable	Yes	Yes	Yes	Yes	Yes	No
Unique *OID*	No	No	No	Yes	No	No
High-level	Yes	Yes/No	No/Yes	Yes	Yes	No

표 1은 네이밍 스킴들의 몇몇의 예에 대한 특징을 보인다. X.500은 ISO 표준 네이밍 서비스이다. 주요 특징은 속성/값 쌍들의 집합으로 구성된다. X.500은 위치 투명성을 제공하며, 읽기 매우 쉽고(Human-readable), 높은 추상화 레벨과, 절대 이름을 제공한다. DNS은 네이밍 호스트와 메일 교환 서버들에 대한 인터넷 표준이다. DNS 이름은 위치 투명성과 읽기 매우 쉬운 특징을 갖는다. 또한 호스트 이름을 인터넷 주소로 매핑시키도록 설계되었으나, 시스템을 찾도록 설계되진 않았다.

URL는 World Wide Web에 대한 기본적인 네이밍 메커니즘으로 이름은 호스트 이름과 파일 이름으로 구성된다. 이는 위치 투명성을 제공하지 않는다. 이와 반면에 Uniform Resource Name은 URL의 위치 투명성 문제점을 해결하고 있다. 하지만 URL만큼 공용화되어 널리 사용되고 있지 않다.

UNIX file 시스템에서 파일들은 때때로 위치 투명성을 제공하고 있다. 분산 파일 시스템으로 이름 공간에 통합되어지며, 마운트가 가능하다. 그러나 위치 서비스에 적용시키지는 않고 있다. 이와 반면에 Stub-Scion Pair Chain(SSP Chain)은 분산 객체에 대한 네임 스킴을 제공한다. 분산 객체의 레퍼런스는 항상 지역 객체 포인터로 객체를 직접적으로 가리키거나 지역 스터브를 통해 원격의 객체를 참조할 수 있다. 이러한 포인터는 읽기 쉽지 않으며, 유일하지 않다. 또한 높은 레벨의 추상화를 제공하지 않는 단점을 갖는다.

2.2 트레이딩 서비스

트레이딩 서비스는 기존 네이밍 서비스와는 달리 속성을 이용하여 객체를 찾아주는 메커니즘으로 네이밍 서비스를 개선한 솔루션으로 보고 있다. 트레이딩 서비스의 상호작용을 위해서 트레이더 객체, 트레이더에게 요청하는 클라이언트 객체(importer) 그리고 트레이더 객체에게 자신의 서비스를 알리는 서버객체(exporter)들로 3가지의 객체가 요구된다. 트레이더 객체는 분산 시스템에서 클라이언트와 서버의 연결을 가능하게 하는 제 3의 객체이며, 특정 서비스에 관한 정보를 서비스 오퍼(Service offer)단위로 관리한다[10]. 서비스 오퍼에는 서비스의 종류 및 그 서비스를 제공하는 인터페이스 레퍼런스, 그 서비스의 QoS(Quality of Service)속성 등을 포함하고 있다.

그림 2에 나타난 바와 같이 이들의 상호작용은 서비스를 제공하는 익스포터가 자신이 서비스할 내용을 서비스 오퍼단위로 트레이더에게 알리면, 트레이더는 그 정보를 자신의 저장소(Repository)에 저장한다. 임포터가 서비스를 받기 위해 트레이더에게 서비스 검색 요청을 하면 트레이더는 익스포트되어 있는 서비스 오퍼를

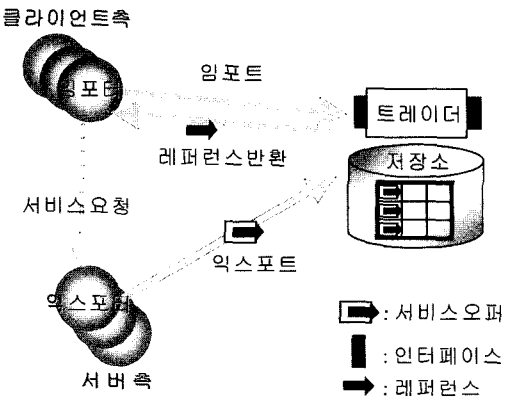


그림 2 트레이딩 서비스의 원리

표 2 다양한 트레이딩 서비스의 특징

Feature	ODP	DRYAD	ANSA	RHODOS	MELODY	TRADE	Y	TOI	DSTC
Classification	Full service	query simple standalone linked	query simple standalone proxy	query simple standalone linked	query simple standalone linked	query simple standalone linked	query simple standalone	query simple standalone linked	query simple standalone linked
Dynamically properties	YES	YES	NO	NO	YES	YES	YES	YES	YES
Service type	YES	YES	YES	YES	YES	YES	YES	YES	YES
Implementation platform	specification	UNIX, Debbie database system	ANSAware system	RHODOS and UNIX	DCE, CORBA, X.500, Melody management system	DCE, DCE directory service	BERKOM environment with admini -strative data repository	CORBA, (Orbix), Postgres	Solaris, Windows NT, object database
Techniques for Improving response time	not applicable	caching, prefetching	none	long-lived imports	long-lived imports, automatic cooperation modes	none	probing, caching	not specified	not specified

검색하여 클라이언트의 요청에 가장 적합한 서버를 찾아 그 서버의 레퍼런스를 임포터에게 반환하면 임포터는 그 서버에게 서비스 요청을 하게 된다.

표 2에서 나타난 바와 같이 다양한 트레이딩 서비스들의 특징을 보면 Y를 제외한 나머지 트레이딩 서비스는 링크 오퍼레이션이나 프록시(proxy) 오퍼레이션을 이용하여 다른 트레이더들과 연합이 가능하다. 동적인 속성은 ANSA와 RHODOS를 제외한 나머지 트레이더에서 제공하고 있으며, 모든 트레이더에서 서비스 타입을 지원한다. 또한 구현 플랫폼 관점에서는 앞서 언급했던 네이밍 시스템을 이용하고 있다. 또한 응답시간을 개선하기 위한 기술적인 측면이 캐싱 기술과 임포트 시에 클라이언트가 지정하는 영속(Long-Lived) 임포트 기술이 지원된다.

2.3 부하 균형화

일반적으로 분산 시스템에서 성능향상의 기본적인 수단으로 부하 균형화 기술을 이용한다. 부하 균형화의 목적은 시스템 상의 이용 가능한 자원에 대해 균일한 이용을 위해 주어진 전략에 따라 작업을 분산하는데 있다. 여기에서 최적의 전략은 서버들과 시스템에서 요청한 작업에 대해 최소의 응답 시간을 갖도록 하는데 있다. 부하 균형화에는 정적 부하균형화(static load balancing)과 동적 부하 균형화(dynamic load balancing) 방법이 있다.

정적 부하 균형화는 각 노드에 주어진 평균 작업의 도착율과 처리율을 바탕으로 노드들의 부하를 균일하게 하는 방식이다. 대기 행렬 이론(queuing theory)에 의하면, 네트워크의 각 노드를 단일 서버의 대기행렬로 모델화한 경우, 프로세스의 도착율이 처리율에 접근할수록 노드의 대기 행렬 길이와 평균 대기 시간은 극단적으로 증가한다. 노드 사이에서 부하를 균형화 시킴으로써 프로세스의 응답 시간을 감소시키고 시스템의 처리율을 증가시킬 수 있다. 정적 부하 균형화는 수학적 해석이 가능한 경우가 많고, 특정 조건 하에서 이론적인 결과나 전체 시스템의 동작을 이해하는데 유용하다. 그러나 실제 시스템에서는 부하가 항상 변경되므로 이에 능동적으로 대처하는 부하 균형화가 필요하다. 이를 동적 부하 균형화라 한다. 정적 부하 균형화에 비해 해석이 어렵고, 시뮬레이션에 대한 평가나 소규모 시스템의 미세 동작을 해석한 연구가 있을 뿐이다.

정적 부하 균형화 전략은 정적으로 고정된 스키마를 이용한 정보를 사용하여 랜덤하게 서버를 선택하여 작업을 주기적으로 할당한다.

동적인 부하 균형화 전략은 부하 정보를 정확하게 측정하기 위해 최신의 정보를 얻도록 고려해야 한다. 이러한 부하 정보가 최신의 내용으로 갱신하는 시간을 최소화하더라도 부하 정보를 전달하는데 네트워크 부하는 증가하게 된다.

본 논문에서 제시하는 통합 바인딩 서비스 모델은 객체들의 이름은 사용자에게 친숙한 이름 사용과 서비스 객체들에 대한 속성 정보를 등록할 수 있는 트레이딩 기법을 통합하고, 광역 객체 컴퓨팅 환경에서 이름이나 속성에 의해 중복된 서비스 객체를 효과적으로 관리하며, 중복된 객체들 중에 최적 조건의 객체를 선정하기 위해 동적인 부하 균형화 전략으로 시스템의 부하 정보(CPU 이용률)를 이용하도록 한다.

3. 통합 바인딩 서비스 모델

많은 객체들로 구성된 광역 시스템은 개별적인 객체들을 참조할 수 있는 구조화된 방법을 필요로 한다. 그러므로 객체의 바인딩은 이름 또는 속성을 컨택 주소로 매핑 해야 하며, 클래스 객체는 클래스 객체의 인스턴스의 초기화와 생성에 의해 바인딩이 이루어진다. 새로운 객체가 생성되면, 이는 객체의 구성요소 중 일부분이며, 이에 접속할 때에는 컨택 주소를 사용한다. 대부분의 분산 시스템에서 네이밍 시스템은 이름과 네트워크 주소의 매핑을 관리한다. 이 방식은 2가지 문제점을 갖는다. 첫 번째는 광역 네이밍 시스템에서 이름 대 주소 바인딩 방법은 변경하기가 어렵다. 이러한 문제점을 해결하기 위해 데이터 캐쉬 기법을 효과적으로 사용하고 있다. 이동 컴퓨팅 환경에서는 일반적으로 객체들의 위치가 수시로 변화되기 때문에 이러한 방식을 이용하더라도 처리하기 어렵다. 두 번째는 대부분의 네이밍 시스템들은 서로 다른 범위에 이름 공간이 분산되어 있으며, 위치에 종속된 이름을 사용한다. 이러한 위치에 종속된 이름들은 이주와 복사 투명성 처리를 매우 어렵게 한다. 분산 객체들은 매번 위치가 변경되거나 복사될 때 추가되고 삭제된다. 이는 트레이딩 서비스도 마찬가지로 적용되는 문제점이다. 또한 이러한 문제점을 해결하더라도 동일한 서비스를 제공하는 객체가 중복되어 있을 경우, 하나의 객체에 집중하는 문제점을 갖게 된다[11, 12, 13, 14]. 본 장에서는 이러한 문제점을 해결하는 모델에 대해 처리 과정과 각 구성요소의 기능에 대해 기술한다.

3.1 통합 바인딩 과정

통합 바이딩 과정은 크게 두 단계로 이루어지며, 첫 번째 단계는 네이밍/트레이딩 서비스에 의해 객체들의 유일한 식별자인 객체 핸들을 얻어오는 단계이며, 두 번째 단계는 위치 서비스에서 네이밍/트레이딩 서비스로부터 얻어진 객체 핸들에 대한 컨택 주소를 얻는 과정이다. 이는 세부적으로 분류하면 4단계로 나뉜다. 다음 그림 3은 이에 대한 처리 과정을 보인다.

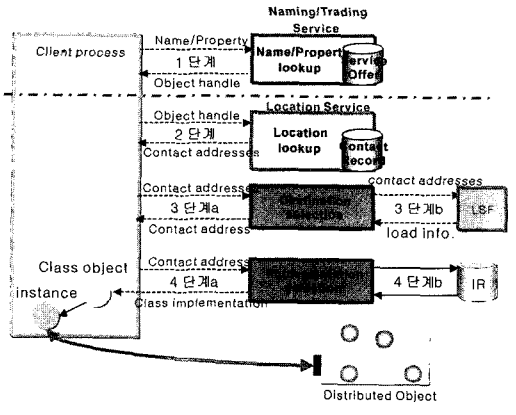


그림 3 통합 바인딩 서비스 처리 과정

위 그림에서 나타난 바와 같이 바인딩 처리 과정은 4단계로 1단계, 2단계는 각각 수행되며, 3단계는 LSF (Load Sharing Facility)의 각 시스템의 부하 정보를 이용하며, 4단계에서는 구현 객체들을 관리하는 IR (Implement Repository)을 이용한다.

-**1 단계** : 이름/속성 검색(Name/Property lookup)
이 단계는 사용자가 정의한 이름 또는 속성을 매개변수로 lookup 메소드 호출에 의해 객체 핸들로 반환된다. 이는 객체의 이름/속성을 객체의 실제 위치로부터 분리됨으로 효과적이다.

-**2 단계** : 위치 검색(Location lookup)
이 단계에서 1단계에서 얻어진 객체 핸들은 객체에 액세스하기 위한 컨택 주소의 집합으로 변환된다. 중복되었을 경우는 객체 핸들당 여러 개의 컨택 주소가 반환된다.

-**3 단계** : 목적지 선정(Destination selection)
객체 핸들과 매핑 되는 컨택 주소의 집합 중에 하나의 컨택 주소를 선택하는 과정으로 해당 객체가 위치한 시스템에 대한 부하정보를 비교하여 최소 부하를 갖는 시스템에 존재하는 객체의 컨택 주소를 반환한다.

-**4 단계** : 구현 객체 선정(Implementation selection)
해당 시스템의 구현 객체를 선택하는 과정으로 실제 구현 객체와 바인딩 하기 위해 클라이언트 프로세스는 새로운 지역 객체를 인스턴스화하고 분산 객체를 초기화한다.

다음은 통합 바인딩 서비스 모델을 이루는 구성요소들에 대해 세부적으로 설명한다.

3.2 네이밍/트레이딩 서비스 구조

광역 객체 컴퓨팅 환경기반에서 기존의 단일 객체의 바인딩은 물론, 중복객체의 바인딩 서비스까지 지원할 수 있는 네이밍/트레이딩 서비스는 이름/속성 별 객체 핸들과 매핑을 제공한다. 이러한 네이밍/트레이딩 서비스는 그림 4와 같이 구성요소를 갖는다.

네이밍/트레이딩 서비스는 서비스를 제공하는 객체를 등록 할 때 유일한 객체 핸들을 부여한다. 객체 핸들은 해당 객체가 생명주기가 끝날 때까지 존재하며, 변경되지 않는다. 하나의 객체가 복사되거나, 이동을 하더라도 동일한 객체 핸들을 갖는다. 그리고 이러한 객체의 세부

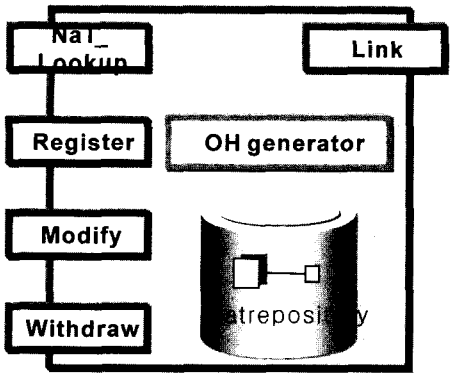


그림 4 네이밍/트레이딩 서비스 구조

적인 정보는 natrepository의 offer_info 테이블과 property_info 테이블에 저장된다.

네이밍/트레이딩 서비스가 제공하는 인터페이스는 5가지 오퍼레이션을 제공하며, 내부적으로 객체 핸들을 생성시키기 위한 오퍼레이션을 갖는다. 이에 대한 세부적인 내용은 다음과 같다.

- NaT_Lookup : 클라이언트의 요구사항을 만족하는 객체의 객체 핸들을 찾기 위한 기능
- Register : 등록할 객체의 서비스를 알리도록 네이밍/트레이딩 서비스가 갖는 저장소에 저장하는 기능
- Modify : 저장소에 저장되어 있는 서비스 오퍼의 내용을 수정하기 위한 기능
- Withdraw : 저장소에 저장되어 있는 서비스 오퍼를 철회하는 기능
- Link : 다른 네이밍/트레이딩 서비스와 연합 할 수 있도록 연결시켜 주는 기능
- Object Handle(OH) generator : 등록되는 객체의 유일한 객체 핸들을 부여하기 위한 기능

다음 그림 5는 네이밍/트레이딩 서비스의 IDL을 나타낸다.

3.3 위치 서비스 구조

위치 서비스는 객체 핸들을 컨택 주소들의 집합으로 매핑을 제공한다. 즉, 위치 서비스의 기능적인 요구사항은 객체 핸들로부터 컨택 주소들의 집합으로 매핑을 제공하는데 있다. 다음 그림 6은 이에 대한 구조이다.

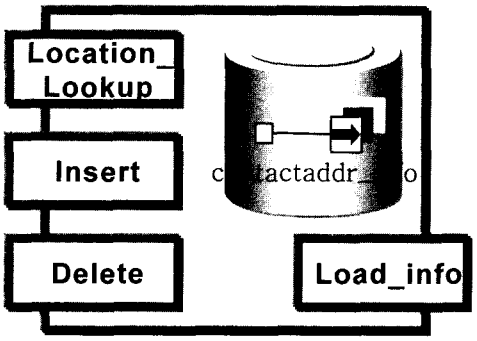


그림 6 위치 서비스 구조

위치 서비스의 인터페이스는 Location_Lookup 오퍼레이션과 시스템의 부하정보를 얻기 위한 Load_info, 그리고 컨택 주소들의 집합을 삽입하거나 삭제하기 위한 Insert, Delete 오퍼레이션을 갖는다. 객체에 대한 위치정보를 관리하기 위한 저장소는 contactaddr_info라는 테이블에서 관리한다.

- Location_Lookup : 객체 핸들에 대한 컨택 주소들의 집합을 탐색
- Insert : 객체 핸들에 대한 컨택 주소들의 집합에 주소 삽입

```
// 네이밍/트레이딩 서비스 오퍼레이션 인터페이스
  interface NaT_Operation
  {
    // 객체 핸들 검색
  ObjectHandle NaT_Lookup(in ObjectName oname, in PropertyValueSeq valueSeq);
    // 객체 등록
  ObjectHandle Register(in ObjectName oname, in ServiceTypeName type, in PropertySeq properties);
    // 객체 수정
    void Modify(in ObjectHandle objhandle, in ObjectName oname, in PropertySeq properties);
    // 객체 철회
    void Withdraw(in ObjectHandle objhandle);
    // 링크
    void Link(in string subpoint);
  };
```

그림 5 네이밍/트레이딩 서비스의 IDL

- Delete : 객체 핸들에 대한 컨택 주소들의 집합에서 주소 삭제
- Load_info : 컨택 주소에 대한 부하 정보 획득 및 최소 부하를 갖는 시스템에 존재하는 객체의 컨택 주소 선정

다음 그림 7은 위치 서비스의 IDL을 나타낸다.

```
// 위치 서비스 오퍼레이션 인터페이스
 interface Location_Operation
 {
   // 컨택 레코드 검색
   conaddrSeq Location_Lookup(in ObjectHandle
            objhandle);
   // 컨택 레코드 등록
   void Insert(in ObjectHandle objhandle, in
            ContactAddr conaddr);
   // 컨택 레코드 삭제
   void Delete(in ObjectHandle objhandle, in
            ContactAddr conaddr);
   // 부하 정보를 참조하여 적절한 컨택주소의 탐색
   ContactAddr Load_Info(in conaddrSeq conaddrs);
 };
```

그림 7 위치 서비스의 IDL

3.4 Load Sharing Facility(LSF)

3.1절에서 언급한 바와 같이 부하 정보는 LSF가 관리하는 시스템들의 부하 정보를 이용한다. LSF는 캐나다의 플랫폼 컴퓨팅사(Platform Computing corporation)에서 개발한 시스템으로 분산된 이기종을 UNIX 및 NT 환경에 관계없이 부하분배와 정교한 배치 스케줄링을 제공한다. 우리가 제시하는 모델에서 위치 서비스는 LSF의 각 시스템의 부하정보를 실시간으로 호출하여 부하가 가장 적은 객체의 컨택 주소를 반환하게 된다. 이에 대한 항목은 다음 표 3과 같다.

다음 (그림 8)은 각 시스템의 부하정보를 획득하여 최소 부하를 갖는 시스템에 존재하는 객체를 선정하기 위한 알고리즘을 보인다.

표 3 부하 정보

항 목	내 용
host_name	LSF에서 사용되는 호스트 이름
status	호스트의 상태
r15s, r1m, r15m	15초/1분/15분 평균 CPU 운행 큐 길이
ut	CPU 이용율
pg	메모리 페이징 비율
ls	현재 로깅한 사용자수
it	UNIX 상의 호스트 유휴 시간
swp	WindowNT상의 swp 공간 이용량
mem	이용 가능한 메모리량
tmp	자유 공간의 양

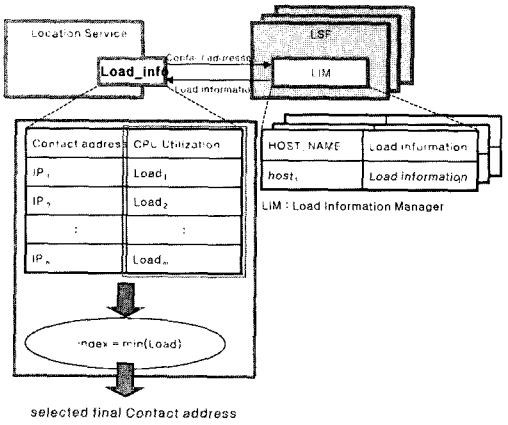


그림 8 위치 서비스에서 부하를 고려한 객체 선정을 위한 알고리즘

네이밍/트레이딩 서비스로부터 얻어진 객체 핸들과 매핑된 컨택 주소들을 각 시스템의 LSF에게 전달하면 LSF의 LIM에 의해 시스템의 모니터링된 부하정보를 위치 서비스에게 반환하게 된다. Load_info 오퍼레이션은 시스템의 상태를 먼저 확인하고, 부하 정보 중에 CPU 이용율만 추출하여 이들 값들을 비교한 후 가장 적은 부하를 갖는 컨택 주소를 선정하여 반환한다.

3.5 광역 통합 트리 구조

통합 바이딩 서비스는 객체의 이주 및 복사를 처리하기 위해 광역 통합 트리를 기반으로 한다. 그림 9에서 나타난 바와 같이 광역 통합 트리 구조는 각 통합 바인딩 서비스들의 연결로 구성되며, 이는 통합 바인딩 서비스들의 관리적인 영역으로 도메인을 이룬다. 따라서, 각 도메인에는 한 개 이상의 네이밍/트레이딩 서비스를 제공하는 서버와 위치 서비스를 제공하는 서버가 위치한다. 각 통합 바인딩 서비스가 위치한 도메인상의 모든 분산 객체에 대한 정보를 갖고 있다. 잎 노드 A22에 서비스를 제공하는 분산 객체에 접근 할 수 있는 컨택 주소들이 저장되어 있다. 노드 RA는 중간 노드로써 하나의 컨택 필드를 갖는 컨택 레코드와 하위레벨을 가리킬 수 있는 연결 정보인 포워딩 포인터에 대한 정보를 갖고 있다. 따라서 클라이언트의 요청은 통합 바인딩 서비스의 포워딩 포인터 정보를 이용하여 하위 노드 또는 상위 노드로 접근하는데 사용된다. 즉, Dom(A1) 내에 클라이언트가 검색 요청이 있을 경우, 도메인 내의 통합 바인딩 서비스가 관리하는 정보를 검색하고 없을 경우에는, 상위 노드인 RA에 대한 포워딩 포인터를 따라 검

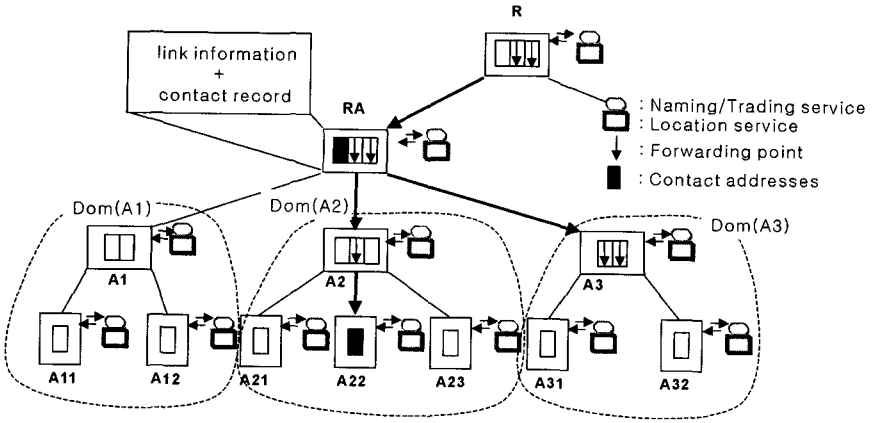


그림 9 논리적인 광역 통합 트리 구조

색 요청을 전달한다. 최악의 경우 근 노드를 경유하여 다른 도메인상의 통합 바인딩 서비스에 요청하게 된다. 통합 바인딩 서비스의 내부적인 절차는 클라이언트 프로세스에게 투명성을 제공한다[13].

4. 구 현

본 장에서는 앞서 제시한 통합 바인딩 서비스에 대한 구현 환경에 대해 기술하고, 설계 내용과 이에 대해 검증을 위해 윈도우 화면으로 네이밍/트레이딩 서비스와 위치 서비스의 오퍼레이션별 수행 결과화면을 보인다. 끝으로 특정 웹 상의 객체를 이름/속성 검색을 통한 바인딩 결과를 보인다.

4.1 구현 환경

구현 환경은 Sun UltraSparc 2i로 333Mhz의 CPU속도에 운영체제로 Sun Solaris 2.7인 시스템 1대와 X86-based PC 2대로 CPU속도는 각각 500Mhz, 400 Mhz이며, 운영체제는 Windows 2000 서버로 구성하였다.

광역 객체 컴퓨팅 환경에는 CORBA 사양을 따르는 상용화된 미들웨어로 Borland의 VisiBroker 4.1을 사용하였으며, 통합 바인딩 서비스의 구성요소인 네이밍/트레이딩 서비스와 위치 서비스는 자바(Java2 SDK 1.3.1)로 개발하였다. 서비스 오퍼와 컨택 레코드 정보를 관리하기 위해 관계형 데이터베이스 시스템인 SQL Server 2000을 이용하였으며, 각 시스템의 부하정보를 획득하기 위해 LSF 4.1을 이용한다.

통합 바인딩 서비스의 구성요소인 네이밍/트레이딩 서비스와 위치 서비스는 하나의 특정 시스템에 위치시키고, 네이밍/트레이딩 서비스가 분산 객체를 관리하는 영역으로 하나의 도메인을 구성하였다. 그리고 분산 객체는 도메인 내에 각 시스템에 배치하였다. 또한 시스템의 부하정보를 얻기 위한 LSF 4.1은 각 시스템에 위치하여 모니터링하여 부하정보를 제공한다.

각 구성요소의 객체들에 대한 인터페이스는 IDL로 작성하였으며, VisiBroker4.1 IDL 컴파일로 컴파일 하였다. 또한, 검증을 위한 그래픽 사용자 인터페이스(GUI)는 자바의 경량 컴포넌트인 JFC/Swing(Java Foundation Class/Swing)을 이용하여 구현하였다. 다음 그림 10은 본 연구의 검증을 위한 구현 환경을 나타낸다.

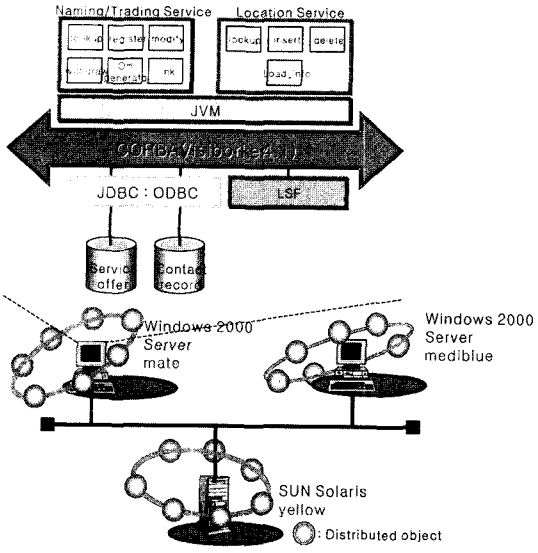


그림 10 구현 환경

4.2 통합 바이딩 서비스 과정

본 절에서는 앞서 보인 검증을 위한 윈도우에서 통합 바인딩 서비스 과정을 네이밍/트레이딩 서비스와 위치 서비스의 다음 표 4에서 나타난 오퍼레이션 별 수행 결과 화면을 보인다.

표 4 통합 바인딩 서비스를 위한 각 서비스 과정

네이밍/트레이딩 Service 과정		위치 서비스 과정	
내용	오퍼레이션	내용	오퍼레이션
서비스 오퍼 등록	Register()	위치 정보 저장	Insert()
객체 검색	NaT_Lookup()	위치 정보 검색	Location_Lookup()
서비스 오퍼 수정	Modify()	위치 정보 삭제	Delete()
서비스 오퍼 삭제	Withdraw()		

4.2.1 서비스 오퍼 등록

분산 객체의 서비스를 알리기 위해 네이밍/트레이딩 서비스가 갖는 저장소에 서비스 오퍼를 저장한다. 네이밍/트레이딩 서비스의 Register 오퍼레이션을 호출하면 Register 오퍼레이션은 OHgenerator를 이용하여 등록되는 객체의 유일한 객체 핸들을 생성하고, 객체 이름, 서비스 타입 이름, 속성들을 저장소에 저장한다.

그림 11은 서비스 오퍼의 당 객체 핸들을 생성하고 서비스 오퍼 정보인 객체 이름, 서비스 타입 이름, 속성 값을 저장소에 저장하는 화면이다.

OHgenerator 오퍼레이션은 네이밍/트레이딩 서비스의 내부 오퍼레이션으로 Register 오퍼레이션이 호출되면 128비트로 이루어진 UUID(Universally Unique IDentifier)을 생성하여 Register에게 값을 넘겨준다.

그림 12는 네이밍/트레이딩 서비스의 Register 오퍼

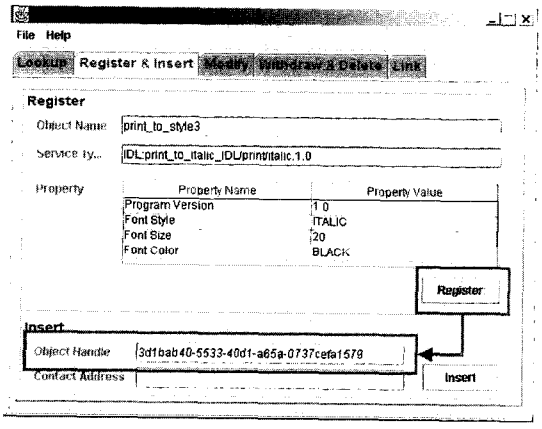
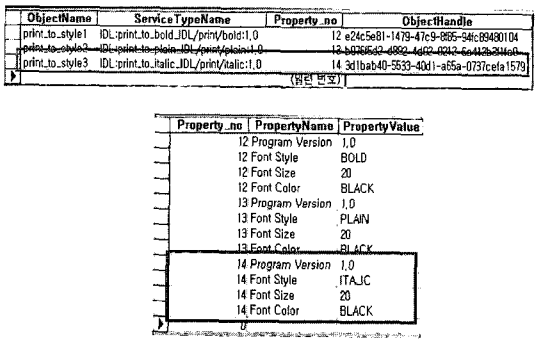


그림 11 서비스 오퍼 등록 및 생성된 객체 핸들 출력 화면



ObjectName	ServiceTypeName	Property_no	ObjectHandle
print_to_style1	IDL:print_to_bold_IDL/print/bold:1.0	12	e24c5e81-1479-47c9-8f85-94fc89480104
print_to_style2	IDL:print_to_plain_IDL/print/plain:1.0	13	b976f5d2-d892-4d92-9212-6c412b3f1fc0
print_to_style3	IDL:print_to_italic_IDL/print/italic:1.0	14	3d1bab40-5533-40d1-a65a-0737cefa1579

(위치 정보)

Property_no	PropertyName	PropertyValue
12	Program Version	1.0
12	Font Style	BOLD
12	Font Size	20
12	Font Color	BLACK
13	Program Version	1.0
13	Font Style	PLAIN
13	Font Size	20
13	Font Color	BLACK
14	Program Version	1.0
14	Font Style	ITALIC
14	Font Size	20
14	Font Color	BLACK

그림 12 네이밍/트레이딩 서비스에 등록된 서비스 오퍼 정보 및 속성 테이블

레이션으로부터 그림 11에서 입력한 값에 따라 해당 저장소에 등록된 결과 테이블이다.

그림 13의 ①은 객체 핸들과 컨택 주소들의 정보를 저장하기 위한 컨택 레코드를 위치 서비스의 Insert 오퍼레이션을 호출하여 저장소에 등록하는 화면이다. ②, ③은 중복객체에 대한 컨택 주소를 등록하는 화면이다.

그림 14는 위치 서비스의 Insert 오퍼레이션으로부

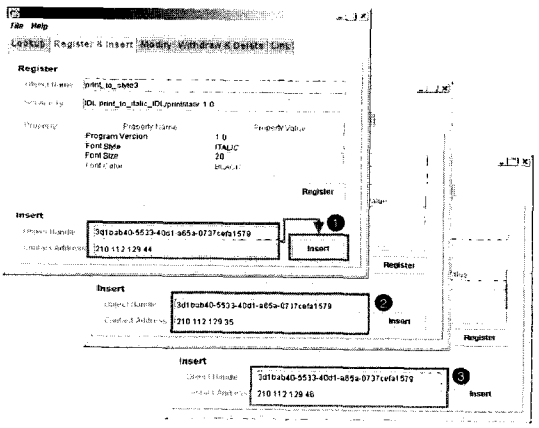


그림 13 중복된 객체의 컨택 주소 등록 화면

ObjectHandle	ContactAddress
e24c5e81-1479-47c9-8f85-94fc89480104	210.112.129.44
e24c5e81-1479-47c9-8f85-94fc89480104	210.112.129.46
e24c5e81-1479-47c9-8f85-94fc89480104	210.112.129.35
b976f5d2-d892-4d92-9212-6c412b3f1fc0	210.112.129.44
b976f5d2-d892-4d92-9212-6c412b3f1fc0	210.112.129.35
3d1bab40-5533-40d1-a65a-0737cefa1579	210.112.129.44
3d1bab40-5533-40d1-a65a-0737cefa1579	210.112.129.46
3d1bab40-5533-40d1-a65a-0737cefa1579	210.112.129.35

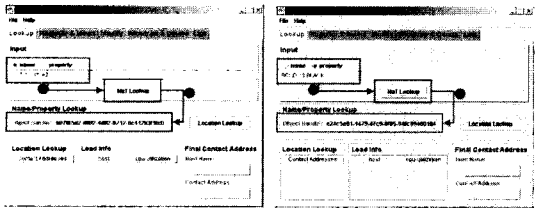
그림 14 위치 서비스에 등록된 컨택 레코드 테이블

터 등록된 컨택 레코드 등록 결과 테이블이다. 동일한 객체 핸들에 서로 다른 컨택 주소는 중복객체가 등록되어 있음을 나타낸다.

4.2.2 객체 검색

객체의 검색은 클라이언트의 요구사항을 만족하는 객체 핸들을 찾아야 하며, 다음으로 찾아진 객체 핸들에 해당하는 한 개 이상의 컨택 주소를 얻어와 컨택 주소에 해당하는 각 시스템의 부하정보를 비교하여 시스템의 부하가 가장 적은 컨택 주소를 선택함으로서 최적 서비스를 제공할 수 있는 객체를 선정한다. 시스템들의 부하 정보 및 최소 부하를 갖는 시스템에 존재하는 객체를 얻기 위해 위치 서비스의 Load_info 오퍼레이션을 사용한다.

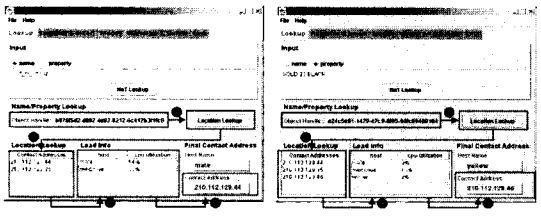
객체 핸들을 찾는 방법으로는 이름으로 검색하는 방법과 속성으로 검색하는 방법이 있다. 그림 15의 (a)는 객체 핸들을 얻기 위하여 이름으로 검색하는 과정을 보이고, (b)는 속성으로 검색하는 과정을 보인다. 각각 이름과 속성 값으로 네이밍/트레이딩 서비스의 NaT_Lookup 오퍼레이션을 호출하는 화면이다.



(a) 이름으로 검색 (b) 속성으로 검색

그림 15 네이밍/트레이딩 서비스의 Lookup 오퍼레이션을 통한 객체 핸들 획득

그림 16은 네이밍/트레이딩 서비스에서 얻어진 객체 핸들을 가지고 이에 해당하는 분산 객체의 실제 위치정보인 컨택 주소들을 위치 서비스의 Lookup 오퍼레이션을 호출하여 탐색한 결과 화면이다.(a)는 이름으로 검색한 결과화면이고 (b)는 속성으로 검색한 결과화면이다.



(a) 이름으로 검색 (b) 속성으로 검색

그림 16 위치 서비스의 Lookup 오퍼레이션을 통한 컨택 주소 획득

그림 16에서 (a)와 (b)의 ③은 객체가 위치한 시스템의 부하 정보를 보여준다. 위치 서비스의 Lookup 오퍼레이션을 통해 획득한 컨택 주소에 해당하는 각 시스템의 CPU 성능을 보여주며, 값이 낮을수록 부하가 적음을 나타낸다. 따라서 부하가 가장 적은 최종 컨택 주소를 (a)와 (b)에서 ④에서 보이고 있다.

4.2.3 서비스 오퍼 수정

서비스 오퍼의 정보는 네이밍/트레이딩 서비스의 Modify 오퍼레이션을 통하여 수정한다.

그림 17은 특정 객체의 이름과 속성을 네이밍/트레이딩 서비스의 Modify 오퍼레이션을 통해 수정하는 화면으로, 속성 값 중 Font Size를 20에서 15로 수정하고, Font Color를 BLACK에서 RED로 수정한 화면이다.

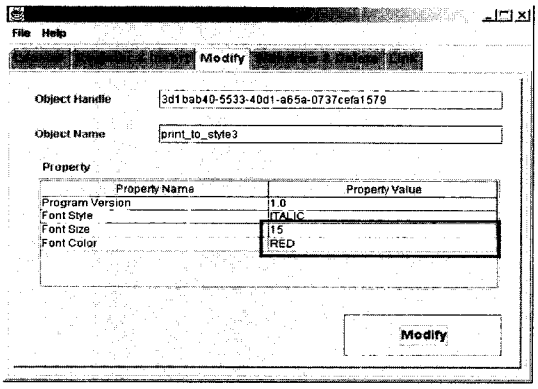


그림 17 네이밍/트레이딩 서비스의 서비스 오퍼 수정 화면

그림 18은 네이밍/트레이딩 서비스의 Modify 오퍼레이션으로부터 수정된 결과 테이블이다. (a)는 수정하기 이전의 속성 테이블로 Font Size가 10이고 Font Color가 BLACK이다. (b)는 서비스 오퍼 수정으로 Font Size가 15로 Font Color가 RED로 수정되었다.

Property_no	PropertyName	PropertyValue
12	Program Version	1.0
12	Font Style	BOLD
12	Font Size	20
12	Font Color	BLACK
13	Program Version	1.0
13	Font Style	PLAIN
13	Font Size	20
13	Font Color	BLACK
14	Program Version	1.0
14	Font Style	ITALIC
14	Font Size	20
14	Font Color	BLACK

(a)수정 전 속성 테이블

Property_no	PropertyName	PropertyValue
12	Program Version	1.0
12	Font Style	BOLD
12	Font Size	20
12	Font Color	BLACK
13	Program Version	1.0
13	Font Style	PLAIN
13	Font Size	20
13	Font Color	BLACK
15	Program Version	1.0
15	Font Style	ITALIC
15	Font Size	15
15	Font Color	RED

(b) 수정 후 속성 테이블

그림 18 서비스 오퍼 수정 결과 속성 테이블

4.2.4 컨택 주소 삭제

위치 서비스가 관리하는 저장소에 저장되어 있는 컨택 주소를 위치 서비스의 Delete 오퍼레이션을 통하여 삭제할 수 있다. 이는 객체의 이동을 지원할 때 사용되거나, 중복된 객체의 컨택 주소를 삭제할 경우 사용된다.

그림 19는 삭제하고자 하는 분산 객체의 컨택 주소를 위치 서비스의 Delete 오퍼레이션을 호출하여 삭제하는 화면이다.

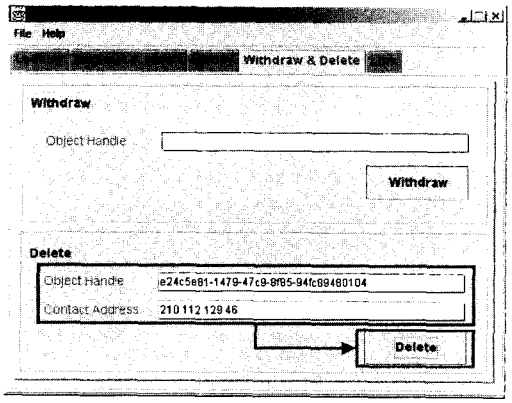
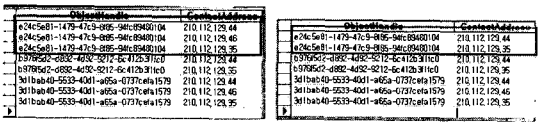


그림 19 컨택 레코드 삭제 화면

그림 20에서 (a)는 컨택 레코드 삭제 이전의 컨택 레코드 테이블이고 (b)는 컨택 레코드 삭제 이후의 컨택 레코드 테이블을 보인다.



(a) 삭제 전 컨택 레코드 테이블 (b) 삭제 후 컨택 레코드 테이블

그림 20 컨택 주소의 삭제 결과 테이블

4.2.5 서비스 오퍼 삭제

네이밍/트레이딩 서비스의 저장소에 저장되어 있는 서비스 오퍼를 네이밍/트레이딩 서비스의 Withdraw 오퍼레이션을 통하여 삭제한다. 이는 등록된 객체를 완전하게 삭제한다는 의미이다.

그림 21은 입력한 객체 핸들에 해당하는 서비스 오퍼를 삭제하는 화면으로 네이밍/트레이딩 서비스의 서비스 오퍼 및 위치 서비스의 컨택 레코드가 모두 삭제된다.

네이밍/트레이딩 서비스가 관리하는 서비스 오퍼의 삭제는 다음 그림 22와 같이 offer_info와 property 테

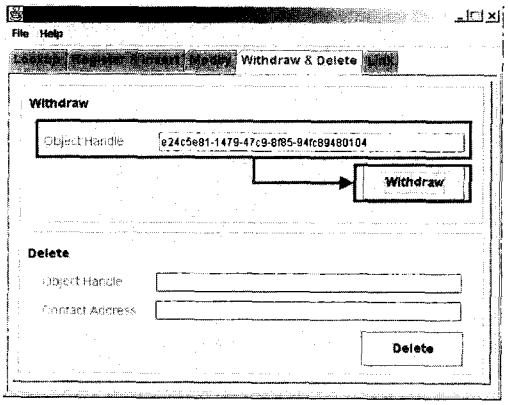
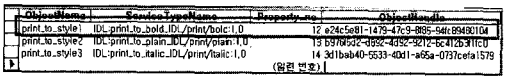
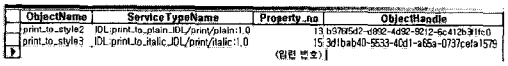


그림 21 서비스 오퍼 삭제 화면



(a) 삭제 전 서비스 오퍼 정보 테이블



(b) 삭제 후 서비스 오퍼 정보 테이블

Property_no	PropertyName	PropertyValue
12	Program Version	1.0
12	Font Style	BOLD
12	Font Size	20
12	Font Color	BLACK
13	Program Version	1.0
13	Font Style	PLAIN
13	Font Size	20
13	Font Color	BLACK
15	Program Version	1.0
15	Font Style	ITALIC
15	Font Size	15
15	Font Color	RED
0		

(c) 삭제 전의 속성 정보 테이블

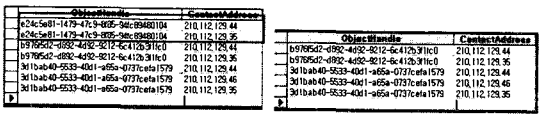
Property_no	PropertyName	PropertyValue
13	Program Version	1.0
13	Font Style	PLAIN
13	Font Size	20
13	Font Color	BLACK
15	Program Version	1.0
15	Font Style	ITALIC
15	Font Size	15
15	Font Color	RED
0		

(d) 삭제 후 속성 정보 테이블

그림 22 서비스 오퍼 삭제 결과 테이블

이블로 (a)와 같이 특정 분산 객체의 서비스 오퍼를 삭제할 경우 (b)에서 나타난 바와 같이 해당 서비스 오퍼 정보가 삭제되었음을 볼 수 있다. 그리고 이와 관련된 속성 정보 테이블에서도 (d)와 같이 삭제된다.

또한, 서비스 오퍼의 삭제는 객체의 완전한 삭제를 의미함으로 위치정보도 함께 삭제되어야 한다. 다음 그림 23은 위치 서비스가 관리하는 컨택 주소들의 삭제된 결과를 보이고 있다.



(a) 삭제 전의 컨택 주소 정보 테이블 (b) 삭제 후의 컨택 주소 정보 테이블

그림 23 서비스 오퍼 삭제로 함께 삭제된 컨택 레코드 결과 테이블

4.3 통합 바이딩 서비스 수행 결과

통합 바이딩 서비스 수행을 검증하기 위해 클라이언트가 통합 바인딩 서비스를 통하여 최소 부하를 갖는 시스템에 존재하는 객체의 컨택 주소를 얻어 서버측에 바인딩 하여 서버로부터 분산 객체인 애플릿을 다운로드하여 클라이언트의 브라우저에 결과를 디스플레이 한다. 이에 대한 과정은 다음 그림 24와 같다.

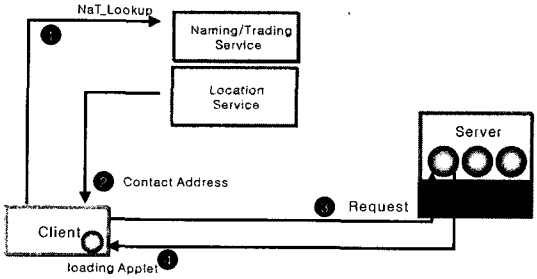
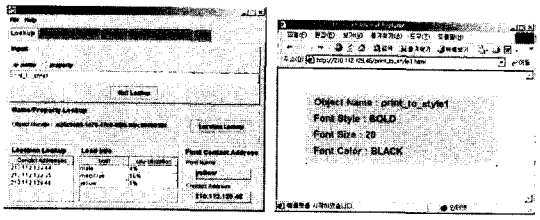


그림 24 통합 바이딩 서비스 수행 과정

다음 그림 25는 검증을 위한 그래픽 사용자 인터페이스인 윈도우에서 NaT_Lookup 탭에서 이름으로 검색하기 위한 화면과 서버의 서비스결과 화면을 보인다.

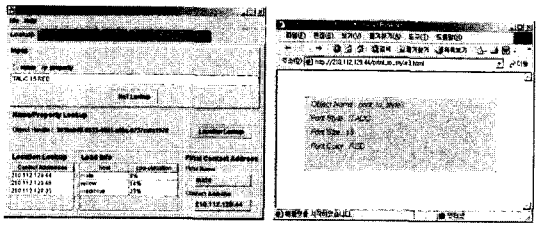
검색할 객체의 이름을 print_to_style1로 검색한 결과 객체 핸들은 'e24c5e81- 1479-47c9-8f85-94fc89480104' 이며 이에 위치 정보는 3개로 중복되어 있음을 알 수 있다. 이들 중 부하 정보가 1%로 가장 적은 yellow의 컨택 주소를 얻어, 해당 서버에게 요청하여 클라이언트의 웹 브라우저에 (b)와 같은 결과를 보이고 있다.



(a) 이름으로 검색 (b) 서비스 결과화면

그림 25 이름으로 검색한 결과 화면 및 서비스 수행 결과 화면

다음 그림 26은 검증을 위한 그래픽 사용자 인터페이스인 윈도우에서 NaT_Lookup 탭에서 속성으로 검색하기 위한 화면과 서버의 서비스 수행 결과 화면을 보인다.



(a) 속성으로 검색 (b) 서비스 결과 화면

그림 26 속성으로 검색한 결과 화면 및 서비스 수행 결과 화면

검색할 객체의 속성을 글자체는 ITALIC이며, 글자크기 15이고, 글자색깔은 RED로 검색한 결과 객체 핸들은 '3d1bab40-5533-40d1-a65a-0737cefa1579'이며 이에 위치 정보는 3개로 중복되어 있음을 알 수 있다. 이들 중 부하 정보가 9%로 가장 적은 mate의 컨택 주소를 얻어, 해당 서버에게 요청하여 클라이언트의 웹 브라우저에 (b)와 같은 결과를 보이고 있다.

5. 결 론

현재 인터넷 기반의 웹서비스와 전자메일과 같은 광역 분산 서비스들이 클라이언트에게 제공되고 있지만, 국부적인 분산 투명성만을 제공하고 있다.

이러한 분산 투명성을 제공하는 솔루션으로 이름 서비스가 대부분을 차지하고 있다. 그러나 이름만으로는 광역 환경에서는 클라이언트의 요구에 맞는 서비스를 제공하기엔 한계가 있다. 이를 해결하기 위해 속성 기반의 트레이딩 서비스가 출현하게 되었다. 이러한 서비스들이 기반이 된 광역 객체 컴퓨팅 환경에서는 수많은

객체들이 이름이나 속성에 의해 중복됨을 예측하게 한다. 그러나 이들은 객체 대 주소 매핑에 의한 바인딩 서비스를 제공하고 있다. 이는 객체의 이동과 중복성을 해결하지 못하는 단점을 갖는다.

따라서, 단일 객체뿐만 아니라 이름 또는 속성 기반의 중복된 객체들의 효율적인 관리와 최소 부하를 갖는 시스템에 존재하는 객체에 대한 통합 바인딩 서비스 모델을 제안하였다. 우리의 모델은 객체의 유일한 식별자인 객체 핸들과 실질적인 액세스를 위한 주소를 나누어, 객체 핸들은 네이밍과 트레이딩이 통합된 네이밍/트레이딩 서비스에서 관리하고, 컨택 주소는 위치 서비스에서 관리하도록하여, 객체의 중복과 이동을 효과적으로 지원한다. 이에 대한 구현 환경은 분산 컴퓨팅 환경에는 CORBA사양을 따르는 VisiBroker 4.1을 이용하였으며, 각 구성요소는 자바(Java2 SDK 1.3.1)로 구현하였다. 또한 부하 정보는 LSF(Load Sharing Facility)의 추출된 정보를 이용하여, 분산 객체의 등록과 최소 부하를 갖는 시스템에 존재하는 객체의 선정을 위한 검색 과정을 보였다.

향후 연구로는 논문에서 제시한 내용을 근거로 연합 구조[10]와 광역 통합 트리구조에서 분산 객체의 탐색 시간에 대한 성능 평가하기 위한 테스트를 해야 하겠다. 또한, 여러 시스템에 분산되어 있는 중복객체들로부터 최소 부하를 갖는 객체를 선정하여 바인딩하기 위한 지능형 모델 기반의 부하분배에 관한 세부적인 연구가 필요하다.

참 고 문 헌

[1] OMG, The Common Object Request Broker : Architecture and specification, Revision 2.0, Tech. report 96-03-04, OMG, July 1995.

[2] K. Obraczka, P. Danzing, and S.-H. Li. "Internet Resource Discovery Services," Computer, 26(9) : 8-22, Sept. 1993.

[3] R.M. Needham. Naming and Security. ACM Press, New York, 1988.

[4] S. Radicati. X.500 Directory Services : Technology and Deployment. International Thomson Computer Press, London, 1994.

[5] P. Mockapetris. "Domain Names-Concepts and Facilities," RFC 1034, Nov.1987.

[6] OMG. OMG RFP5 Submission: Trading Object Service May 1996. OMG Document orbos/ 96-05-06. http://www.omg.org/docs/orbos/96-05-06.ps

[7] G. Ballintijn, M. van Steen, A.S. Tanenbaum. "Exploiting Location Awareness for Scalable Location-Independent Object IDs," Proc. Fifth Annual ASCI Conf., Heijen, The Netherlands, pp. 321-328, June 1999.

[8] M. van Steen, F.J. Hauck, G. Ballintijn, A.S. Tanenbaum. "Algorithmic Design of the Globe Wide-Area Location Service," The Computer Journal 41(5):297-310, 1998.

[9] Elarbi Badidi, Rudolf K. Keller, Peter G. Kropf, Vincent V. Dongen, "The Design of a Trading-based COBRA Load Sharing Service," In Proceeding of the Twelfth International Conference on Parallel and Distributed Computing Systems(PDCS' 99), pp. 75~80. 1999.

[10] 정창원, 주수종, "객체그룹간의 상호접속을 지원하는 연합 트레이더 모델", 한국정보과학회 논문지, 제 26권 9호, pp. 1126~1134, 1999.9.

[11] 전병택, 정창원, 주수종, "광역 분산 객체들의 바인딩 지원을 위한 연합 네이밍/트레이딩 모델", 2001년 춘계 학술발표 논문집, 제 28권 1호, 정보과학회. pp. 427~429, 2001.4.28.

[12] 전병택, 정창원, 주수종, "광역 객체 컴퓨팅 환경에서 분산 객체의 관리를 위한 서비스 모델의 설계," 추계 학술발표 논문집, 제 8권 2호, 정보처리학회. pp. 309~312, 2001. 10.13.

[13] 전병택, 정창원, 주수종, "광역 객체 컴퓨팅 환경에서 분산 객체의 통합 바인딩 서비스의 최적 객체 선정", 춘계 학술발표 논문집, 제 9권 제 1호, 정보처리학회. pp. 1499~1502, 2002. 4.12.

[14] 정창원, 오성권, 주수종, "광역 객체 컴퓨팅 환경에서 이름/속성기반의 통합 바인딩 서비스 방안", 제 9권-A권 제2호. 정보처리학회 논문지. pp. 241~248, 2002. 6.

정 창 원

1993년 원광대학교 컴퓨터공학과 졸업(학사). 1998년 원광대학교 컴퓨터공학과 졸업(교육석사). 1998년~현재 원광대학교 컴퓨터공학과 박사과정. 관심분야는 분산 컴퓨팅, 분산 객체 모델, 위치 서비스, 트레이딩 서비스, 멀티미디어 데이타베이스

오 성 권

1981년 연세대학교 전기공학과 졸업(학사) 1983년~1989년 금성산전연구소(선임연구원). 1993년 연세대 대학원 전기공학과 졸업(박사). 1996년~1997년 캐나다 Manitoba대학 전기 및 컴퓨터공학과 Post-Doc 1993년~현재 원광대 전기전자 및 정보공학부 부교수. 2002년~현재 대한전기학회(KIEE) 및 제어자동화시스템공학회(ICASE) 편집위원. 관심분야는 시스템 자동화, 퍼지이론 및 신경회로망 응용, 계산지능 및 소프트 컴퓨팅

주 수 종

1986년 원광대학교 전자계산공학과(학사). 1988년 중앙대학교 대학원 컴퓨터공학과(석사). 1992년 중앙대학교 대학원 컴퓨터 공학과(박사). 1993년~1994년 미국 Univ. of Massachusetts at Amherst, 전기 및 컴퓨터공학과, Post-Doc. 1990년~현재 원광대학교 컴퓨터 공학과 교수. 2002년 12월~현재 미국 Univ. of California at Irvine 전기공학 및 컴퓨터과학과 연구교수. 관심분야는 분산실시간 컴퓨팅, 분산 객체 모델, 시스템 최적화, 멀티미디어 데이타베이스