濟州大學校 産業技術硏究所 論文集, 第10卷, 2號, pp.99-107, 1999. 12
Cheju Nat'l Univ. Res. Insti. Ind.Tech. Jour., Vol.10, No.2 pp.99-107, 1999. 12

JNDI를 이용한 Mobile Agent 기반 망관리시스템

김 정 철* · 김 강 석** · 송 왕 철***

Network Management System Based upon Mobile Agent using JNDI

Jung-Chul Kim*, Gang-Suk Kim** and Wang-Cheol Song***

ABSTRACT

Recently, network management technology by delegation has been studied as a kind of network management technology to satisfy the user's requirements. Mobile Agent in the technology issues in the distributed network system. Mobile Agent is a software program which is able to move on network with mission for managing a NE(Network Element). Therefore, We have studied the efficient network management technology which is the network management system based upon Mobile Agent Using JNDI(Java Naming Directory Interface). JNDI is API(Application Program Interface) to support interface which make user's application program to be used the naming & directory service. We have designed and implemented the mobile agent network management system using JNDI. This Mobile Agent system is expected to provide a solution for the conventional centralized network system.

Key words : JNDI, mobile agent, naming & directory service, network management.

I. 서 론

오늘날 컴퓨터와 통신망의 급속한 발달은 우리 사회에 많은 영향을 미치고 있다. 뛰어난 성능을 가진 컴퓨터들이 망에 접속되어 있으며, 이를 이용해 우리는 서로의 정보를 주고받으며 공유하기도 한다. 특히 클라이언트/서버 구조를 갖는 분산시스템은 시스템 간 자원의 공유가 가능하며 작업 수행 성능 향상에 기여하였다. 클라이언트/서버 개발에 있어 단일 서버용 솔루션만 다루던 과거와 달리 이제는 인터넷은 물론 다중망 운영체제 및 다중 플랫폼을 이용해 네트워크 중심의 분산 애플리케이션을 구축하고 있다. 이런 네트워크관리를 위해서는 IERF의 SNMP (Simple Network Management Protocol) 프로토콜과 OSI의 CMIP(Common Information Management Protocol)이 있다. 기존의 중앙집중식의 SNMP를 사용한 망관리시스템은 관리자가 관리대상의 대리자로부터 SNMP 메시지를 전송하고 전송 받으며 관리자는 대리자로부터 전송 받은 메시지를 분석하여 망관리를 수행하게 된다.[1] 이러한 중앙집중형 망관리 모델은

* 제주대학교 대학원
Graduate School Cheju Nat'l Univ.
** 제주관광대학
Cheju Tourism Collage
*** 제주대학교 정보공학과, 산업기술연구소
Dept. of Informatioin Eng., Res. Insti. Ind. Tech., Cheju Nat'l Univ.

간결한 망 구성 및 관리가 가능하고 보안이 용이하며, 망 관리에 적합한 모델이기는 하나, 중앙의 관리자에 모든 관리 기능이 집중되어 있기 때문에 망 확장성의 제한, 실시간 관리 기능의 제한, 서비스의 동적인 추가의 어려움 등의 문제점을 가지게 되었다.[2] 이러한 문제점들을 해결하기 위하여 SNMP표준안에서 계층적 관리모델 등을 제시하기는 하였지만 별다른 효과가 없었다.[3]

Mobile Agent는 분산환경에서 최근 부각되고 있는 연구분야이며 Mobile Agent 고유의 자율성, 비동기성, 이동성 등은 망관리에 적용하기 적합한 특성으로 Mobile Agent를 망관리에 적용함으로 인하여 중앙집중식의 망관리의 난점들을 해결할 수 있다.[3]

네이밍 & 디렉토리 서비스는 분산환경에서 객체를 참조하거나 객체의 속성을 참조하여 객체를 찾을 수 있도록 하는 서비스이다.[4] 본 논문에서는 이 네이밍 & 디렉토리 서비스를 이용하여 관리자가 Mobile Agent를 파견하고자 하는 관리대상을 참조할 수 있도록 JNDI(Java Namaing Directory Interface)를 이용하였으며, 관리자가 네이밍 서비스와 디렉토리 서비스를 받아 Mobile Agent를 생성 또는 파견할 수 있도록 하는 시스템을 설계하고 구현하였다.

II. 분산망 관리

분산망관리를 위해 가장 널리 쓰여지는 프로토콜인 SNMP의 중앙집중식 망관리의 난점은 Mobile Agent 기술을 망관리에 접목시킴으로서 많은 부분이 해결될 것이다.

2.1. SNMP기반 망관리

TCP/IP 네트워크 환경에서 가장 많이 사용되고 있는 관리방법인 SNMP 관리시스템은 관리스테이션, 관리대리자, MIB (Management Infomation Base), 네트워크 관리 프로토콜로 구성되어 있다. 관리스테이션은 데이타 분석 및 장애복구 위한 프로그램, 네트워크의 현황을 관찰할 수 있는 인터페이스, 원격에 있는 네트워크 요소의 관리 능력, MIB를 통한 획득한 정보의 데이타베이스화가 기본기능이다. 호스트, 브리지, 라우터 및 허브 등에 관리대리자가 구동되어, 긴급사항 또는 관리스테이션의 요구에 응답할 수 있다. 네트워크 관리프로코콜에 의해 관리스테이션 관리대리자 사이의 링크를 형성하는 주요 기능은 다음과 같다.

- Get : 관리 스테이션이 관리대리자를 통해 객체의 값을 조회.
- Set : 관리 스테이션이 관리대리자를 통해 객체의 값을 세팅.
- Trap: 에이전트가 특별 이밴트를 관리 스테이션을 보고.

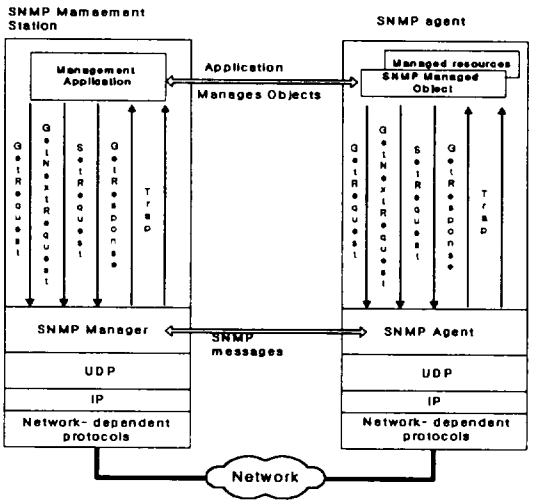

Fig. 1 Role of SNMP

SNMP는 어플리케이션 레벨 프로토콜로서 UDP (User Datagram Protocol)상에서 동작하도록 구성되어 있다. Fig. 1 에서 보는 것같이 관리스테이션이 GetRequest, GetNextRequest, SetRequest의 유형으로 메세지를 보내면 대리자의 응답은 GetResponse에 의해 수행된다.

2.2. Mobile Agent 기반 망관리

Agent는 사용자를 대신하여 사용자가 원하는 어떤 일을 수행해주는 프로그램이라고 할 수 있다. 또한 Mobile Agent는 네트워크 환경을 이동하면서 관리기능을 수행 할 수 있는 프로그램을 의미한다. 이동 에

이전트 시스템은 원격 시스템의 코드를 호출하는 기존의 클라이언트/서버 개념, 클라이언트의 요청이 있을 때 코드를 보내주어 실행하는 방식인 Code-on-demand 방식과는 달리 Mobile Agent가 네트워크를 통해 해당시스템으로 전송되어 실행되는 방식이며 자신의 판단에 의해 이동한다. NMS(Network Management System)에서 Mobile Agent가 해당 네트워크 장비로 직접 이동해서 수행되고, 관리정보를 획득하고 원하는 NE(Network Element)로 이동 할 수 있기 때문에 중앙집중형 망관리 모델에 비해 다음과 같은 특징을 가진다.

- NMS의 부하 분산 : Mobile Agent가 직접 NE로 이전 되어 NE에서 오퍼레이션을 하여 그 결과 값만을 가지고 오므로 NMS의 부하 감소.
- 확장성 제공 : 네트워크 오버헤드를 감소시켜 망관리 범위를 확대.
- 실시간 관리의 용의 : 에러진단, 긴급복구 등의 실시간의 요구되는 망관리 경우 대처능력의 탁월.
- 서비스 추가의 용이 : 새로운 서비스를 능동적으로 대처.
- 망연동의 유연성 : 서비스의 동적인 추가로 서로 이질적인 타 망과의 연동에 유연.
- 효율적인 망 운용 : 각각의 NE에 맞는 망관리 기능을 이동 에이전트를 통해 감시함으로서 NE의 특성을 충분히 살리는 최적화 된 망관리가 수행되고, 불필요한 MO(Managed object) 요소가 감소.

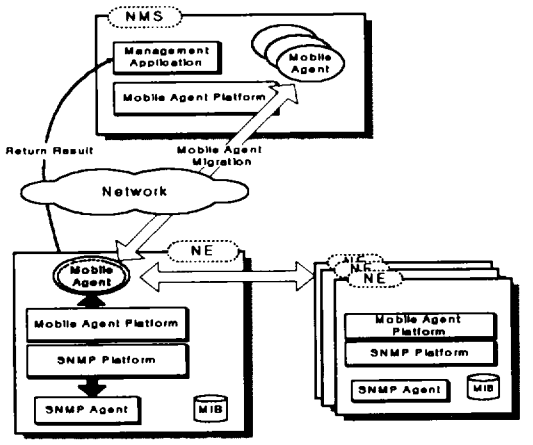

Fig. 2 Mobile agent on network

Fig. 2은 이동 에이전트의 활동 구조를 나타내고 있다.

III. JNDI

본 논문에서는 관리자가 대리자의 정보를 직접 참조하여 Mobile Agent를 파견 또는 이동시킬 수 있도록 하기 위하여 JNDI를 이용하여 네이밍 & 디렉토리 서비스를 사용한다.

3.1. 네이밍 서비스

네이밍 서비스는 분산된 객체의 객체 참조를 쉽게 얻어올 수 있도록 객체 참조를 한곳에 모아 참조할 수 있다. 사용자 관점에서는 혼합된 네임을 구성하는 namespace가 있는 것이다. 즉, 객체의 이름을 가지고 분산된 객체를 쉽게 찾기 위한 객체를 이름과 함께 등록하고, 클라이언트는 네이밍 서비스 서버에 접속하여 등록된 이름을 사용해서 객체에 대한 객체 참조를 얻는다.

각 객체들은 네이밍 서버에 자신의 정보를 등록하고 클라이언트는 네이밍 서비스 서버에 접속하여 등록된 이름을 사용하여 객체에 대한 참조를 얻는다. 네이밍 서비스은 대부분 디렉토리 서비스로 확장되어진다.

네이밍 서비스는 등록된 객체를 그림과 같이 계층적인 트리 형태로 관리하게 된다. 이러한 계층적인 트리 구조를 네이밍 그래프(Naming Graph)라 한다.

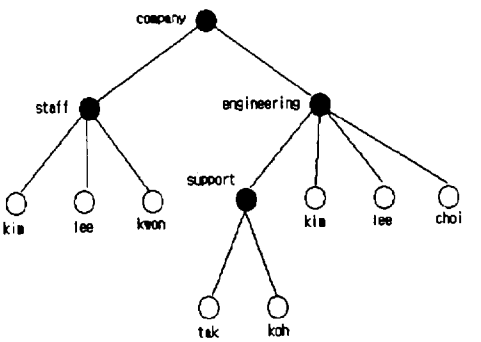

Fig. 3 Naming graph

Fig. 3의 네이밍 그래프에서 각각의 사원은 다음에 서처럼 컨텍스트와 객체이름으로 구성된 자기만의 합성된 객체이름을 가지게 된다. Fig. 3과 같이 staff라는 부서에 일하는 kim라는 사원은 자신만의 독특한 합성이름인 company.staff.kim 라는 합성이름(Compound Name)을 가지게 된다.

Naming Service의 예를 들어보면

1) SNS(Simple Name Service)

SNS(Simple Name Service)는 하나의 작은 조직내에서 사용하기 위해 설계되여졌다. 예를 들면 한 개의 기업, 학교, 한정된 조직 등에서 사용자, 컴퓨터, 서비스에 대한 명명할 수 있다. 이 경우는 한정된 지역 뿐만아니라 사용자 또한 권한이 인증된 사용자만이 이용할 수 있다.

2) DNS(Domain Name System)

사람이 인터넷의 규모와 호스트의 증가에 따른 IP주소만으로는 인식이 어려워 새로운 명명체제가 필요하여 등장하게 되었다. 인터넷에 연결된 특정 컴퓨터의 Domain name을 IP Address를 찾아 변환해 주는 일을 하는 컴퓨터 체계는 모든 인터넷 사용자에게 서버이름을 분석할 수 있도록 도와주는 디렉토리 서비스기능도 제공함으로써 수년간 역할을 훌륭히 수행해 왔지만 하나의 함수로 제한되어 있다는 단점이 있다.

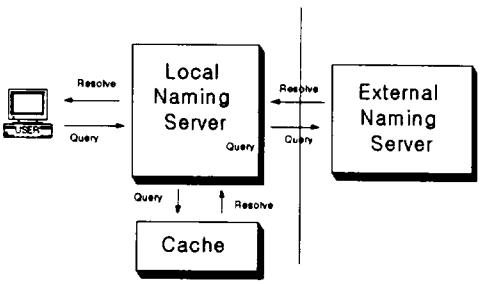

Fig. 4 Domain name system architecture

Fig 4.는 DNS는 사용자 프로그램은 Local Name Server에 Query를 하면 Naming Sever의 프로세스는 먼저 Cache에 질의를 한 후 그 해당 name 이름이 존재하면 그에 대한 IP를 Resolve 하여주고 만약에 없다면 외부 Name Server에 다시 질의를 하는 과정을 갖는다. 이때 외부 Name Server는 사전에 Local Name Server에 등록되어 있는 외부루트서버로서 먼저 사용자의 질의한 Name을 이 외부루트서버(Root Name Server)에 질의를 한다.

3.2. Directory Service

Directory 서비스는 사용자가 네트워크상에서 지능적으로 자원을 찾을 수 있는 애플리케이션을 구축할 수 있도록 한다. 좋은 예로 전자우편의 애플리케이션은 사용자그룹의 위치를 알아내고 워드프로세싱은 애플리케이션은 프린터를 찾아내며 클라이언트/서버 애플리케이션은 자원들 네트웍 어디에 존재하는지 관계없이 데이터 베이스를 발견한다. 애플리케이션 객체는 어느 특정 서버상에 위치하는 것이 아니라 모든 네트워크상에서 존재한다. 따라서 개발자가 객체의 위치를 파악하기 위해 일반 Infrastructure를 확보하는 일은 필수적이라 할 수 있다. 객체와 데이터는 모두 Directory 서비스에 등록되어 있거나 디렉토리 상주하고 있다.

현재의 네트워크 Directory 서비스에는 DNS, 노벨의 네트웨어 디렉토리 시스템과 노벨 디렉토리서비스, 넷스케이프의 디렉토리 서버, 마이크로소프트의 액티브 디렉토리와 CCITT의 X.500 등이 있다.

1) X.500

1988년 국제전신전화자문위원회(CCITT)는 X.500 디렉토리 서비스를 출간하였다. X.500은 세계적인 디렉토리 정보의 보고로 사용자 이름과 패스워드, 사진, 위치, 액세스 컨트롤 메카니즘 등 귀중한 자료를 보유하고 있다. 애플리케이션이 자원을 쉽게 찾는 메카니즘을 제공하는 X.500은 DNS보다 적응하기 쉽고 기능과 특징이 더욱 다양하다는 장점을 갖는다. X.500 Directory 모델은 정보의 논리적 데이터베이스 제공을 돕는 시스템 중 하나로 분산된 집단 시스템이다.

X.500 DIT(Directory Informtion Tree)에 있는 모든 엔트리는 진정한 속성들의 모음이다. X.500은 국가, 지역, 조직, 조직단위, 사람, 별명 등의 객체 클래

스를 정의한다.[1]

2) LDAP(Lightweigh Directory Access Protocol)

SMTP(Simple Mail Transfer Protocol)이 E-mail 송수신의 표준이고, HTTP가 인터넷상의 도큐먼트 전달을 위한 표준이 듯이, LDAP은 디렉토리 참조를 위한 표준이다. 기술적으로, LDAP은 RFCs 1777과 1778에 의해 TCP/IP상의 온-라인 와이어 비트 프로토콜(on-line wire bit protocol)로 정의되었다. LDAP 프로토콜은 X.500 DAP(Directory Access Protocol)로의 자원요청 없이 X.500 모델들을 지원하는 디렉토리 엑세스를 제공하기 위해서 제작되었다. 이 프로토콜은 특별히 디렉토리들 상호간의 읽기/쓰기 엑세스를 제공하는 처리응용(Management Application)들과 브라우저 응용들을 목표로 삼았다. 이것은 X.500을 위한 보완물로서 사용되어진다. 세션계층과 프리젠테이션 계층을 우회하여 TCP나 다른 전송계층 프로토콜의 지랫대 역할을 수행 할 수 있다. X.500을 이해 디렉토리 모델 엔트리에 기초하고 있으며 엔트리에 언급된 이름과 구별되는 이름을 사용하고 있다. 반면 고도로 구조화된 X.500데이터 인코딩 메카니즘을 사용하는 대신에 LDAP는 엔트리를 표시하기 위해 단일한 문자열에 기초한 접근 방식을 시도하고 있다. 따라서 애플리케이션의 Directory 서비스 요구는 문자열이 LDAP는 엔트리가 객체 클래스 속성을 사용하게 함으로써 세분화된 질의로 검색을 할 수 있어 디렉토리 데이터베이스의 검색방법을 훨씬 더 용이하게 했다.

3) 넷스케이프 디렉토리 서버

Netscape Directory Server는 LDAP를 충실하게 구현하였다. 즉, 제품 자체가 LDAP를 위해 디자인되었다. 넷스케이프는 넷스케이프 커뮤니케이터에서 LDAP를 지원하고 있다. 대부분의 디렉토리들은 각각 몇몇 특정 디렉토리 작업을 위해 최적화되어 있다.

X.500은 DAP를 위해 최적화 되어 있고, 노벨 Directory Server는 디렉토리 지원 NetWare 응용 프로그램을 위해 최적화되어 있으며, Lotus Notes는 LotusScript 콜을 위해 최적화 되어 있다. 이 디렉토리들은 게이트웨이라고 불리는 트랜잭션 엔진을 통해 LDAP 디렉토리를 지원해야한다. 트랜잭션 엔진를 통한 LDAP으로의 지원은 성능 저하, 전체 시스템에 대한 견고성 저하(게이트웨이 자체가 시스템 장애 포인트가 될 수 있음), 프로토콜 비매치 문제 발생에 대한 잠재성, 관리의 복잡성 가중 등의 문제를 초래한다. 반면, 처음부터 순수한 LDAP 디렉토리로 제작된 Netscape Directory Server는 게이트웨이를 필요로 하지 않는다.

넷스케이프 디렉토리 서버는 윈도우 NT 뿐만아니라, 솔라리스, HP-UX, Linux, 등 몇몇 플랫폼 상에서도 유용하다. 관리자는 모든 넷스케이프 서버에 있는 HTTP 기반 Administration Server를 이용하여 Directory 서버를 관리 할 수 있다. Directory 서버는 관리자가 다양한 지리적 위치에서 마스터 서버상에 있는 장치들을 조직화 할 수 있도록 하는 중복 매카니즘을 제공한다.

3.3. JNDI

JNDI(Java Naming Directory Interface) 네이밍 & 디렉토리 서비스를 제공하는 Java API(Application Program Interface)이다. JNDI는 선 마이크로시스템에 의해 개발되었으며 JNDI를 사용하므로서 자바 애플이케이션은 자바 객체를 저장하고 검색할 수 있다. 게다가 JNDI은 어트리뷰트(attributes)을 사용하는 객체을 검색하거나 객체에 그 어트리뷰트를 연결하는 것 같은 기본적인 디렉토리 오퍼레이션을 수행하기 위한 방법을 제공한다. JNDI는 네이밍 & 디렉토리 서비스 구현에 독립적이다. 이것은 공통된 API를 사용하여 각양각색의 네이밍 & 디렉토리 서비스를 자바응용프로그램을 통해 엑세스할 수 있도록 하여준다.

여러 네이밍 & 디렉토리 서비스 제공자는 공통된 API 뒤에서 단락없이 접속될 수 있다. 이것은 Java 응용프로그램이 LDAP, NDS, DNS, NIS 그리고 CORBA 같이 존재하는 다양한 네이밍 & 디렉토리 서비스를 이용할 수 있도록 해준다. JNDI을 사용하므로서 자바 응용프로그래머는 강력하고 인식성이 강한 응용프로그램을 만들 수 있다. 그 응용프로그램이 채택되어 환경과 잘 접합될 수 뿐만 아니라 자바의 객체 모델의 이점을 이용 할 수 있다.

Fig. 5는 JNDI 구조를 보여 주고 있다.

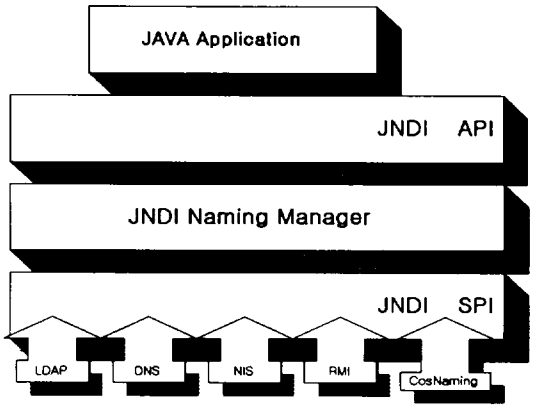

Fig. 5 Architecture of JNDI

- javax.naming : The Naming Interface 제공
- javax.naming.directory : The Directory Interface 제공
- javax.naming.event : The Event Interface 제공
- javax.naming.ldap : The LDAP Interface 제공
- javax.naming.spi : The Service Provider Interface 제공

3.4. JNDI를 이용한 Mobile Agent 망관리

Mobile Agent 기반 망관리는 앞 절에서 언급한 것과 같이 관리자가 자신의 역할을 Mobile Agent에 일임하여 직접 대리자가 있는 관리대상시스템에 파견하고 Mobile Agent 는 일임 받은 임무를 수행하여 관리자의 역할을 대신하는 구조를 가진다. 관리자가 Mobile Agent를 생성하고 파견하는 것에 대해 기존의 방법들은 관리자가 대리자를 선택하는 부분에 있어 어떠한 참조 또는 정보를 제공받지 못하였으며 분산환경에서 망의 투명성을 제공받지 못하였다. 본 논문에서 제안한 구조에서는 관리자는 네이밍 혹은 디렉토리 서버로부터 네임 혹은 속성들을 참조할 수 있기 때문에 분산환경에서 중요하게 여겨지는 망투명성 중 위치투명성을 제공받을 수가 있다.

Ⅳ. 시스템 설계 및 구현

JNDI를 이용하여 네이밍 & 디렉토리 서비스를 Mobile Agent 기반 망관리에 제공하기 위하여 제안한 구조와 구현결과는 다음과 같다.

4.1. 시스템 구조

Fig. 6은 본 논문의 Mobile Agent 기반 망 구조에 대하여 나타내고 있다. 망관리를 하기위해 관리자는 JNDI를 이용한 미리 갖추어진 네이밍 서버로부터 DB로부터 이동 할 네트워크요소(NE)의 위치(IP)와 NE의 속성등을 참조 받아 대상 해당 NE에 Mobile Agent를 파견시킬 수 있다. Mobile Agent는 NE의 대리자와 SNMP 오퍼레이션을 수행하여 관리자의 작업을 대신하고 결과를 관리시스템에게 보고한다.

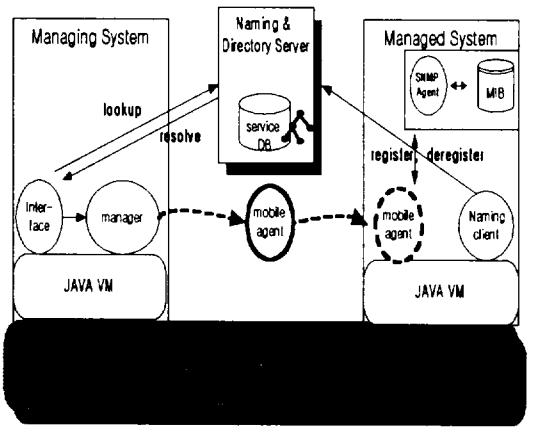

Fig 6. Architecture of supposed system

4.2. 시스템 구현 환경

구현한 시스템의 환경은 다음과 같다.

- Managed System의 환경
  - 하드웨어 : x86계열 Intel Pentium
  - 운영체제 : 솔라리스2.7.
- Managing System의 환경
  - 하드웨어 : SUN 워크스테이션(SUN Ultra-2).
  - 운영체제 : 솔라리스 2.6.
- Naming & Direcoty server 의 환경:
  - 하드웨어 : x86계열 Pentium Ⅱ
  - 운영체제 : Windows NT
- 시스템 구현에 사용된 언어(도구)

- 2 SDK v1.2(JAVA 언어)
- JAVA beans(AdventNet SNMPv1 API class)
- Voyager ORB 3.1
- Netscape Directory server 4.1

구현한 시스템은 Fig. 7과 같은 순서로 동작하며 각각의 과정은 다음과 같다.

피관리시스템이 JNDI서버에게 자신의 정보를 등록시킨다(①). 관리시스템의 응용프로그램은 Mobile Agent를 생성하여(②) Directory Server에 등록된 Mobile Agent는 등록된 관리대상시스템의 정보를 네이밍 서비스를 받아(③·④) 관리대상시스템으로 이동하여(⑤) Mobile Agent에 부여된 의무는 SNMP 대리자와 오퍼레이션을 통해(⑥). SNMP 대리자는 MIB-Ⅱ 값을 가져와 관리시스템으로 넘긴다(⑦).

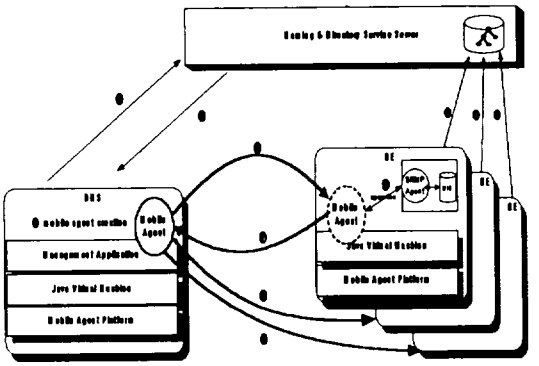

Fig. 7 System operation procedure

4.3. Global Naming Tree

본 논문에서는 LDAP Directory Server를 통해 Namig Service를 하기 위해 객체를 Fig. 8 같은 트리 구조로 객체를 등록하였다. 이러한 트리를 Global Naming Tree이라 한다. 객체대상으로는 각 시스템의 SNMP 대리자를 설정하였다. 그 SNMP 대리자의 이름을 다음과 같이 CN(Common Name)를 붙여 indigoSNMP, corbaSNMP, InfoworkSNMP, Infocom-SNMP라는 이름으로 등록시켰다. 본 구현 시스템에서는 CNU를 Directory Root로 하여 설정하였기 때문에 DN(Distinguish Name)의 O=cheju.ac.kr를 루트로 등록을 하였다.

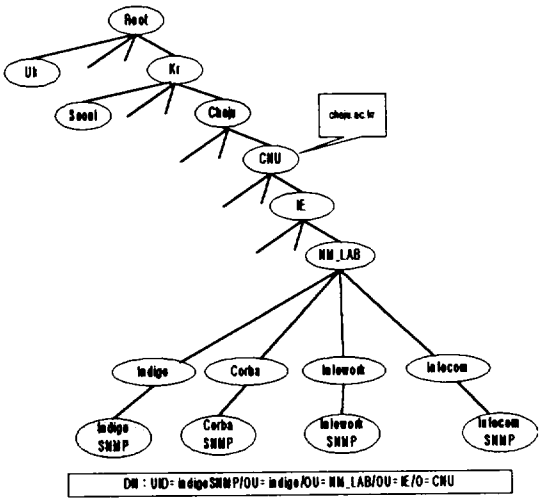

Fig. 8 Global naming tree

4.4. Naming & Directory Service 스키마 구성

본 논문의 Naming & Directory 서버의 객체의 스키마를 위해 다음과 같은 오퍼레이션으로 객체의 등록 및 삭제, 검색 및 조회를 하고 있다.

- Register : Server에 객체를 등록
  ADD([in]Name DN,
  [in]SuperName SuperDN,
  [in]MyType ObjectType
  [in]Host HostInfo);
- Deregister : Server에서 객체를 삭제
  Delete([in]Name DN);
- Search : 객체의 속성으로 조회
  Search([in]MyType ObjectType,
  [[out]Host HostInfo]*);
- LookUp : 객체의 이름으로 조회
  LookUp([in]Name RDN,
  [out]MyType ObjectType,
  [out]Host HostInfo);

4.5. 시스템 구현 및 결과

Fig. 9는 Directory Server에 indigoSNMP 라는 객체를 등록시키는 과정을 보여주고 있다.

Fig. 10은 각 관리대상시스템을 Naming & Directory Server에 등록시킨 후 그 중 indigo대상시스템의 객체

indigoSNMP의 속성(attribute)을 보여주는 화면이다.

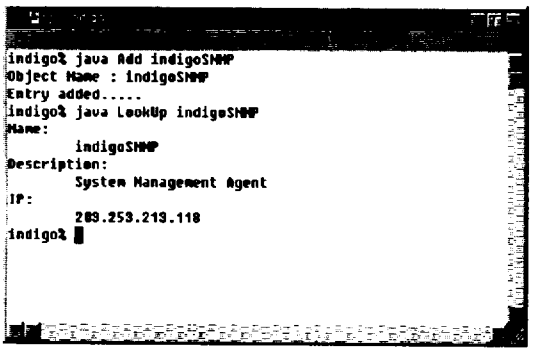

Fig. 9 Registering object of NE

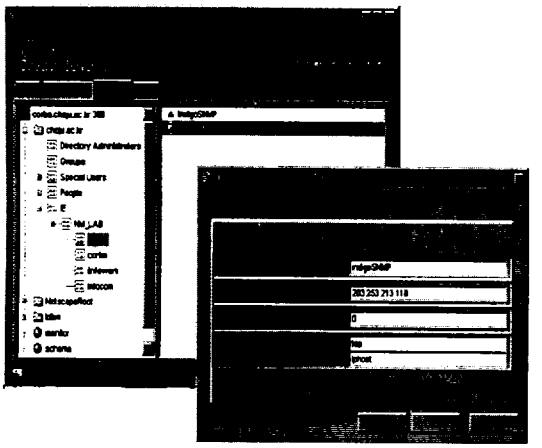

Fig. 10 Displaying object's attribute in directory server

Fig. 11는 관리자가 indigoSNMP라는 관리대상시스템을 관리하기 위해 Naming & Directory Server에게 질의를 하여 해당 관리대상시스템이 Mobile Agent가 이동할 수 있는 IP값을 인수로 받는다. IP값을 받은 Mobile Agent는 그 NE으로 이동하여 시스템 Interface와 시스템의 Network Interface정보를 수집하여 그 값을 관리시스템으로 보낸다.

V. 결 론

본 논문에서는 Mobile Agent가 관리대상시스템에

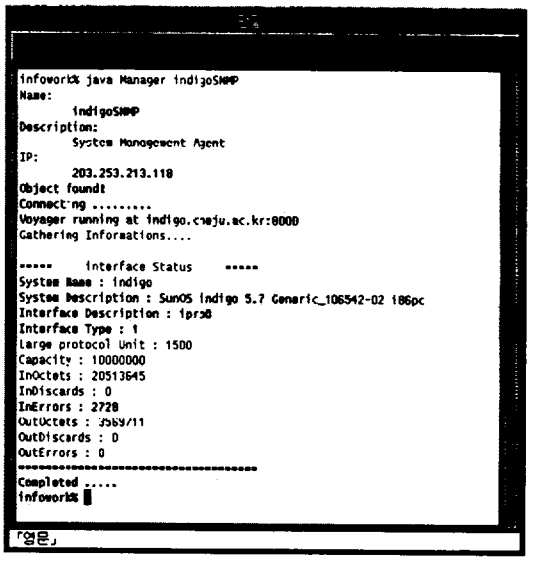

Fig. 11 Display of NMS

존재하는 네트워크 객체에 쉽게 접근하기 위해 네이밍 & 디렉토리 서비스를 이용하는 방안을 제시하였다. 네이밍 & 디렉토리 서비스를 통해 관리하고자 하는 대상시스템의 존재를 확인 할 수 있고, Mobile Agent가 이동 될 대상시스템의 정보를 네이밍 & 디렉토리 서비스를 통해 얻을 수 있어 관리자가 가장 먼저 확인해야 단계인 Voyager 플랫폼이 형성된 관리대상시스템을 네이밍 & 디렉토리 서비스에서 해결하여 주었다.

그 결과로 관리자는 대리자에 대한 세밀한 정보 혹은 망 특성과 위치 특성에 따른 부분들을 고려하지 않아도 됨에 따라 망관리에 위치투명성을 제공할 수 있었으며 이것은 분산환경에서 무척이나 유리한 점으로 작용한다.

관리 기능의 추가 기능으로 Directory 서비스 기능 중 객체를 참조하는데 있어서 이름뿐만 아니라 객체의 속성값을 가지고 객체의 정보를 얻을 수 있는 메카니즘이기 때문에 Agent가 이용 할 데이터 베이스의 구축이 당면 과제로 남아 있고, Mobile Agent가 OMG(Object Management Group)에서 표준화가 진행중인 MASIF(Mobile Agent System Interoperability Facilites)에 부합되는 ORB와의 통합이 연구과제로 남아 있다. 이런 문제가 해결이 되면 Mobile Agent를

이용한 망관리 모델은 더욱더 효과적인 망관리 모델이 될 것으로 고려된다.

참고문헌

1) William Stllings. 1996. SNMP SNMPv2 and RMON. Addison-Wesley. pp.84-120. pp.145-196.
2) 조일권. 1997. 이동에이전트를 이용한 망관리방법에 관한 연구. 석사학위논문. pp.37.
3) R. J. Post. 1995. Manager-Manager MIB analysis and implementation. University of Twente TIOS. pp.67.
4) Danny B. Lange. Mitsuru Oshima. 1998. Programming and Deploying Java Mobile Agents with Aglets. Addison-Wesley. pp.1-30.
5) 이인화. 1994. Corperate LAN상에서 최적의 네트워크 관리 구현 방안. 석사학위논문. pp.4-6.
6) 정보통신부. 1993. 단순망 관리 규약 (snmp) 표준. pp.8-37.
7) Sun Microsystem. inc.. 1999. Java Naming and Directory Interface Application Program Interface (JNDI API. SPI).
8) 왕창종 · 이세훈. 1999. Inside CORBA3 프로그램밍. 도서출판 대림. pp.285-316.
9) 데이터베이스 월드 inc.. 1998. 디렉토리서비스 향배. 58호 pp.87-91.
10) Morris Sloman.1996. Network and Distributed System Management. pp.250-280.
11) Netscape inc.. 1999. "Netscape Directory Server version4.1 Installation Guide". "Server Administrator Guide". "Deployment Guide". "Managing Server with Console".
12) Advent Network Management inc..1996. Advent SNMP Package.
13) ObjectSpace inc.. 1999. Voyager ORB3.1 Developer Guide.
14) 김정철 · 송왕철. 1999. JNDI를 이용한 Mobile Agent 기반 망관리시스템. 한국해양통신학회추계발표집. pp.406-410.