碩士學位論文

JNDI를 이용한 Mobile Agent 기반 네트워크 관리 시스템

濟州大學校 大學院

情報工學科

金 正 哲

1999年 12月

JNDI를 이용한 Mobile Agent 기반 네트워크 관리 시스템

指導敎授 宋 旺 暾

金 正 哲

이 論文을 工學 碩士學位 論文으로 提出함

1999年 12月

金正哲의 工學碩士學位 論文을 認准함

審査委員長 ㄱ 光 ㄱ

委 員 곽 호 영

委 員 송 왕 철

濟州大學校 大學院

1999年 12月

Network Management System Based upon Mobile Agent using JNDI

Jung-Chul Kim

(Supervised by professor WangCheol Song)

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF INFORMATION ENGINEERING
GRADUATE SCHOOL
CHEJU NATIONAL UNIVERSITY

1999. 12.

<목 차>

summary

At present, most computers included personal computer are connected with a network, The users of the computer want to receive various services through network more and more. A network manager has made an effort to meet users' requirement with efficient network management technology. The network management means not only supplying network services but also managing a network itself.

Mobile Agent is a kind of network management technology by delegation, is a software program and is able to move to manage a Network Element on a network.

In this thesis, the Mobile Agent technology and the Simple Network Management Protocol(SNMP) is adopted. The Mobile Agent is serviced by the naming & directory service, which is interfaced by Java Naming Directory Interface(JNDI). JNDI is Application Program Interface(API) to supply interface. The naming & directory service is a essential network service to search or lookup some objects which exist somewhere on a network, even if the managing system has only the objects' name or attribute value. The service is very useful in establishing network-transparency on a network system.

Therefore, This thesis proposes the network management system based upon Mobile Agent by using the naming & directory service. This network management technology is expected to provide a solution for management of the conventional distributed network system.

Ⅰ. 서 론

오늘날 컴퓨터와 통신망의 급속한 발달은 우리 사회에 많은 영향을 미치고 있다. 뛰어난 성능을 가진 컴퓨터들이 네트워크에 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 IETF(Internet Engineering Task Force)의 SNMP(Simple Network Management Protocol) 프로토콜과 OSI(Open Systems Interconnection)의 CMIP (Common Management Information Protocol)이 있다. 본 논문에서 사용한 SNMP관리 프로토콜은 관리자가 관리대상의 대리자로부터 SNMP 메시지를 전송하고 관리자는 대리자로부터 전송 받은 메시지를 분석하여 네트워크관리를 수행하는 방식을 취한다.(Sloman, 1996). 이 SNMP관리 프로토콜에 의해 관리되는 네트워크관리모델은 중앙집중형관리, 계층형관리, 분산형관리로 크게 나눌 수 있다. 중앙집중형 네트워크 관리는 간결한 네트워크 구성 및 관리가 가능하고 보안이 용이하며, 네트워크 관리에 적합한 모델이기는 하나, 중앙의 관리자에 모든 관리 기능이 집중되어 있어 여러 가지 문제점이 나타났다. 예를 들어 네트워크 확장성의 제한, 실시간 관리기능의 제한, 서비스의 동적인 추가의 어려움 등의 문제점을 가지게 되었다. 이러한 문제점들을 해결하기 위하여 SNMP표준안에서 계층적 관리모델 등을 제시하기는 하였지만 큰 호응을 얻지는 못했다. 근래에 들어와서, 분산

형 관리모델이 일반적인 추세이다.(조, 1998).

지역적으로 산재되어 확장된 네트워크관리를 위해 인간이 직접 관리하는 경우도 있지만 많은 시간과 비용이 낭비되어 관리기능을 대신할 무엇이 필요로 하였다. 그것이 소프트웨어 에이전트(Software Agent)이다. 이 소프트웨어 에이전트는 관리시스템 안에서 정적으로 존재하지 않고 관리대상시스템에게로 이동하여 관리대상시스템에서 어떤 오퍼레이션을 하도록 되어있다.(Vu Anh Phamn 등 1998) (Danny B. Lange 등 1998). Mobile Agent는 고유의 자율성, 비동기성, 이동성을 갖고 있어 네트워크 관리에 적용하기 적합하여 중앙집중식의 네트워크관리의 난점들을 해결할 수 있다.

본 논문에서는 이 Mobile Agent에 Naming and Directory 서비스를 추가하여 분산환경에서 객체의 이름이나 속성값으로 객체의 위치를 찾아 Mobile Agent가 이동할 NE(Network Element)를 쉽고 빠르게 찾을 수 있어 네트워크 관리에 효율을 기할 수 있도록 하였다. Mobile Agent가 Naming & Directory service를 받기 위해 JNDI(Java Naming Directory Interface)라는 API를 사용하였다. JNDI을 사용하므로서 여러 종류의 Naming & Directory service (LDAP, CosNaming, NIS, DNS 등)를 사용할 수 있어 미래의 서비스 확장에 대비할 수 있다. JNDI를 사용한 Mobile Agent는 관리하고자 하는 NE의 IP와 Type를 서비스 받아 네트워크의 위치투명성을 확보 할 수 있었다.

본 논문의 구성은 II장에서는 이 논문의 기반이 되는 SNMP기반 네트워크를 설명하기 위해 분산네트워크관리 SNMP 관련 개념과 Mobile Agent 기술을 설명하고, III장에서는 본 논문에서 다루고자하는 내용인 Naming & Directory 서비스와 JNDI에 대해 설명하며, IV장에서는 본 논문에서 제안된 시스템의 구조, 구현 및 결과에 대해 설명하였고, 그리고 마지막장에서는 결론을 맺는 절차로 구성하였다.

Ⅱ. 분산네트워크 관리

효율성과 생산성을 최대화하기 위해 복잡한 네트워크를 통제하는 것으로서 모든 상황에서 각 네트워크 요소들이 제대로 동작하고 최대로 성능을 수행 할 수 있도록 하는 모든 기술 및 관리 활동을 네트워크관리라 한다. 네트워크관리 프로토콜 중 SNMP관리 프로토콜은 네트워크 관리에 가장 널리 쓰여지는 프로토콜이며 대부분 중앙집중식으로 사용되고 있다. 중앙집중식 관리에는 여러 가지 난점이 있는데 이것은 근래 분산환경에 적절한 기법으로 부각되고 있는 Mobile Agent 기법으로 많은 부분이 해결 될 것이다.

1. SNMP기반 네트워크관리

SNMP 관리프로토콜은 TCP/IP 네트워크 환경에서 가장 많이 사용되고 있는 관리프로토콜로서, SNMP 관리시스템은 관리스테이션, 관리대리자, MIB(Management Information Base), 네트워크관리 프로토콜로 구성되어 있다. 관리스테이션은 데이타 분석 및 장애복구 위한 프로그램, 네트워크의 현황을 관찰 할 수 있는 인터페이스, 리모트에 있는 네트워크 요소의 관리 능력, MIB를 통한 획득한 정보의 데이타베이스화가 기본기능이다.(정보통신부, 1993) 호스트, 브리지, 라우터 및 허브 등에 관리대리자가 구동되어, 긴급사항 또는 관리스테이션의 요구에 응답할 수 있다. 관리대상 객체의 현황을 데이터 변수로 표현한 MIB는 관리스테이션을 위해 관리대리자에서 객체를 액세스하는 접근점으로

MIB값의 변경을 통해 Agent의 동작대상 및 구성변경을 수행할 수 있다(조, 1999).

네트워크관리 프로토콜에 의해 관리시스템과 SNMP대리자 사이의 링크를 형성하는 주요 기능은 다음과 같다.

- Get : 관리 스테이션이 관리대리자를 통해 객체의 값을 조회.
- Set : 관리 스테이션이 관리대리자를 통해 객체의 값을 세팅.
- Trap : Agent가 특별 이벤트를 관리시스템에 보고.

SNMP는 에플리케이션 레벨(Application Level) 프로토콜로서 UDP (User Datagram Protocol)상에서 동작하도록 구성되어 있다. Fig 1. 에서 보는 것같이 관리스테이션이 GetRequest, GetNextRequest, SetRequest의 유형으로 메세지를 보내면 Agent의 응답은 GetResponse에 의해 수행된다.(조,1999) (william, 1996) (정보통신부, 1993)

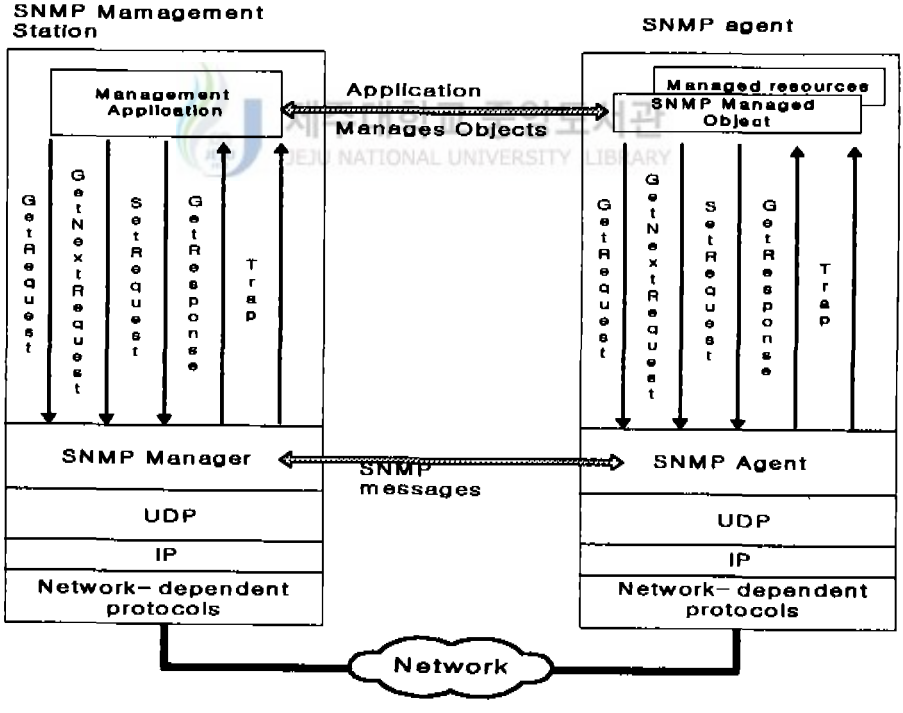

Fig. 1 The Role of SNMP

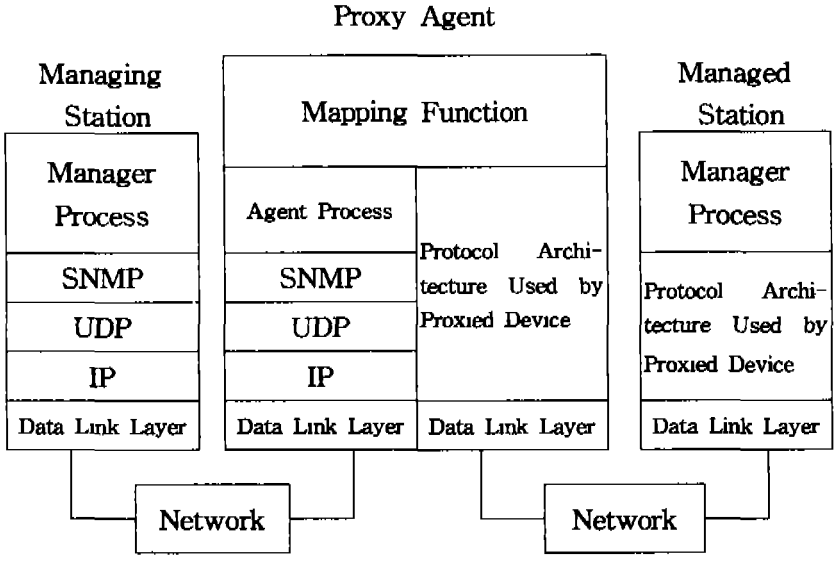

Fig. 2 Proxy Agent

SNMP 대리자 기능을 구현할 수 없는 장비관리를 위해서, SNMP대리자가 이러한 장비의 프록시 에이전트(Proxy Agent) 역할을 수행하는 것으로, 관리스테이션이 해당 장비에 관계된 프록시 에이전트에게 요구하면, 프록시 에이전트는 해당 장비가 사용하는 관리 프로토콜로 변환하여 장비로 보내고, 장비로부터 수신된 정보를 관리스테이션에 되돌려 보내는 것으로 Fig. 2에서 보는 것과 같다.

2. Mobile Agent 기술

Agent란 사전적으로 '대리인', '대행자' 란 뜻을 가지며, 이것은 처음에는 인공지능에서 인간을 대행하는 컴퓨터 프로그램에 관한 연구

분야로서 이 기술에 이동코드(Mobile Code) 개념이 결합하면서 Mobile Agent가 생겨나기 시작하였다. 컴퓨터 분야에서는 사용자를 대신하여 사용자가 원하는 어떤 일을 수행해 주는 프로그램으로 정의 내릴 수 있다.

1) Mobile Agent

중앙집중형 관리구조는 중앙에 관리스테이션을 두고 각 대상스테이션에 다수의 대리자를 두어 각 대리자가 중앙의 관리자의 관리요구에 따라 동작하는 형태로 구성된다. 이런 중앙 네트워크관리구조는 네트워크의 확장이 어려우며, 서비스의 변경 및 추가에 문제점을 갖고 있고, 각 스테이션의 정보를 중앙에 호출해야 하므로 실시간 관리가 어렵다. 그리고 중앙의 관리자가 네트워크 전체의 책임을 지게되어 중앙의 관리자의 부담이 너무 커지는 단점을 갖고 있다. 이러한 단점을 극복하기 위해 제시된 방법으로서 분산관리기법이 근래에 들어 강조되고 있다. (조,1997) 분산네트워크 관리는 우선 관리기능을 네트워크에 분산시켜 놓고 있다. 그리고 플랫폼에 독립성을 갖춘 분산객체기술을 이용하여 네트워크 투명성을 확보하고 있다. 그밖에 다양한 방식의 관리시스템이 연구중에 있으며 그 중 위임에 의한 관리방식(Management by Delegation)으로서 COD(Code On Demand), REV(Remote Evaluation), MA(Mobile Agent)가 있다. 그 중 Mobile Agent는 네트워크를 이동하면서 Agent 기능을 수행 할 수 있는 프로그램을 의미한다. Mobile Agent 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. Mobile Agent는 TeleScript, Java, Perl, TCL 언어로 작성되고

Interpreter로 수행된다. Mobile Agent는 독립적이고 자율적으로 이벤트가 발생했는지 여부를 볼 수 있고, 인터넷상에서 사용자가 원하는 정보가 어디에 있는지 돌아다니면서 탐색할 수 있고, 여러 가지 서비스를 종합적으로 처리할 수 있다.

Mobile Agent는 세가지 주요한 요소를 가지고 있다.

- Agent 프로그래밍 언어 : 응용개발자가 Agent와 Agent가 방문할 위치(플레이스)를 프로그램하기 위한 언어.
- Agent 실행환경 : 언어를 위한 가상기계를 제공하며 Agent와 플레이스 동작을 관리.
- Agent 프로토콜 : Agent끼리 서로 다른 컴퓨터 시스템간에 통신을 가능하게 함.(마, 1999)

Mobile Agent는 해당 관리대상시스템으로 직접 이동해서 수행되고, 관리정보를 획득하고 원하는 관리대상시스템으로 이동 할 수 있기 때문에 중앙집중형 네트워크관리 모델에 비해 다음과 같은 장점을 가진다.

- 관리시스템의 부하 분산 : Mobile Agent가 직접 관리대상시스템으로 이전되어 관리대상시스템에서 오퍼레이션을 하여 그 결과 값만을 가지고 오므로 관리시스템의 부하 감소.
- 확장성 제공 : 네트워크 오버헤드를 감소시켜 네트워크관리 범위를 확대.
- 실시간 관리의 용이 : 에러진단, 긴급복구 등의 실시간의 요구되는 네트워크관리 경우 대처능력의 탁월.
- 서비스 추가의 용이 : 새로운 서비스를 능동적으로 대처.
- 네트워크연동의 유연성 : 서비스의 동적인 추가로 서로 이질적인 타네트워크와의 연동에 유연.

· 효율적인 네트워크 운용 : 각각의 관리대상시스템에 맞는 네트워크관리 기능을 Mobile Agent를 통해 감시함으로서 관리대상시스템의 특성을 충분히 살리는 최적화 된 네트워크관리가 수행되고, 불필요한 MO(Managed Object) 요소가 감소

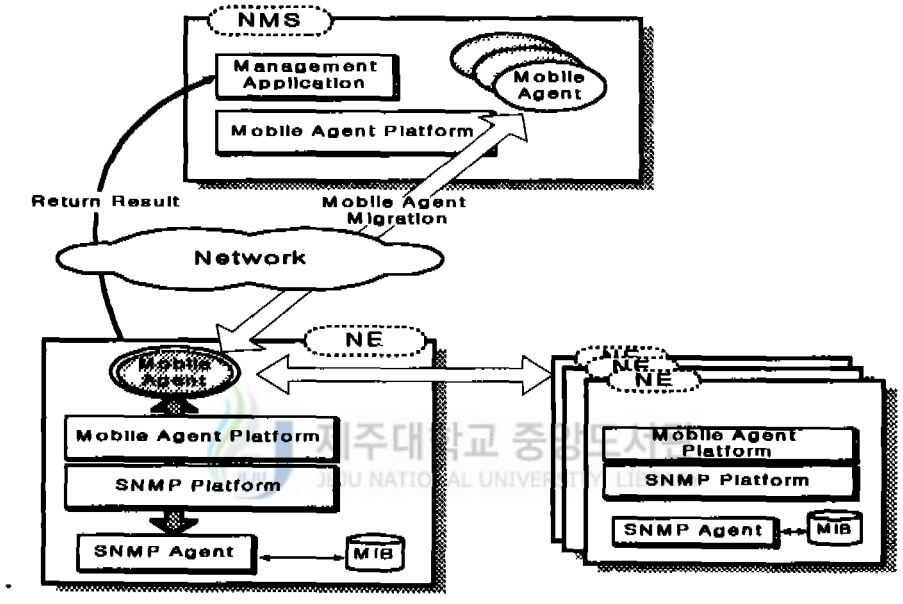

Fig. 3 Network Management Architecture
Based on Mobile Agent

Fig. 3은 Mobile Agent의 활동 구조를 나타내고 있다. 먼저 Mobile Agent가 이동 할 하얀색 화살표로 표시된 바와 같이 관리대상시스템(NE)으로 이동하고, Mobile Agent가 필요로 하는 MIB의 정보를 로컬에 위치해 있는 NE의 SNMP대리자에게 요구한다. 그에 대한 응답 메시지를 받아 NE의 상태를 판단하게 되고, 이상이 있을 경우에 SNMP대리자에게 요구 메시지를 보내 관리대상시스템을 제어하여 적절한 조치를 할 수 있게 된

다. 또한 현 상태나 수행결과를 관리시스템에게 통보할 수 있고, 경우에 따라서는 또 다른 관리대상시스템으로 이동할 수 있다.(조,1997)

2) Mobile Agent Platform

Mobile Agent의 플랫폼은 여러 종류가 있다. 미츠비시사의 Concordia, IBM의 ASDK(Aglets), 제너널맥직의 Odyssey, IKV++의 Grasshopper, ObjectSpace사의 Voyager 등이 있다. Concordia는 Java 기반으로 WWW와 연동이 될 수 있고, IBM의 ASDK는 인터넷 환경에서 Aglets 생성하며, General Magic의 Odyssey는 Mobile Agent를 위한 Java 클래스를 제공하며, TCL을 확장한 스크립트 언어인 Agent-TCL을 이용하는 Dartmouth 대학의 D'Agents, CORBAg 환경에서 MASIF(Mobile Agent System Interoperability Facilites)를 반영한 Grasshopper, 분산환경에서 Java를 이용해서 Mobile Agent를 생성하는 ObjectSpace사의 Voyager 등이 있다. 본 논문에서 채택한 ObjectSpace의 Voyager의 특징은 다음과 같다.

① 순수 자바를 사용한 Agent ORB(Object Request Broker).
Voyager는 처음부터 Object의 Mobility를 지원하도록 개발되었으며 이러한 이유로 Agent나 Object들이 다른 시스템으로 이동이 가능하다. Voyager는 순수 자바 문법을 사용하여 원격지의 자바가상머신을 통해 Object나 Agent와 통신을 할 수 있으며, 이동성이 있어 네트워크 사이를 이동하면서 계속적인 실행을 할 수 있다. 또, 원격지에서 실행을 위한 다른 자바 클래스들의 수정이 필요 없으며 상대적으로 작은 사이즈로 구성이 되어있어 다른 ORB(Object Request Broker)에 비해 빠른 응답시간을 가질 수 있다.

② 클라이언트/서버 방식과 Agent를 이용한 분산프로그래밍 기법을 동시에 지원

Voyager는 네트워크에서 다른 분산환경들(CORBA, DCOM, RMI)과 완벽하게 동작하여 Agent가 네트워크에서 이동성이 보장받아 IIOP(Internet Inter-ORB Protocol)상의 CORBA, ORPC (Object Remote Procedure Call)상의 DCOM(Distributed Component Object Model) 그리고 JRMP(Java Remote Method Protocol)상에서 동작하는 RMI(Remote Method Invocation)등의 이질적인 ORB와도 투명하게 동작이 가능하여 융통성 있는 구조를 가진다. 즉, 사용자가 응용프로그램 작성에 유리하다.

③ Messaging계층에서 단방향 또는 동시 단방향과 같은 다른 형태의 Message Type를 위치와 객체의 모델에 관계없이 객체에 보낼 수 있다.

④ 단일 API를 통해 여러 Naming 서비스 Provider의 사용이 가능하다.

Ⅲ. JNDI

본 논문에서는 관리자가 대리자의 정보를 직접 참조하여 Mobile Agent를 파견 또는 이동시키는데 있어서 네트워크의 위치투명성을 제공하기 위해 Naming & Directory 서비스를 사용하였다.

1. Naming서비스

Names이란 컴퓨터, 서비스, 포트, 각 객체의 정보와 같은 폭 넓게 다양한 자원을 참조 및 사용하기 위한 것으로서 분산 네트워크관리시스템에서는 자원의 공유 및 통신을 위해 반드시 필요로 한다. 이러한 Names을 분산네트워크시스템에서는 개방성을 위해 분산된 객체의 객체 참조를 쉽게 얻어올 수 있도록 객체 참조를 한곳에 모아 참조할 수 있도록 한다. 사용자 관점에서는 혼합된 네임을 구성하는 Namespace가 있는 것이다. 즉, 객체의 이름을 가지고 분산된 객체를 쉽게 찾기 위한 객체를 이름과 함께 등록하고, 클라이언트는 Naming & Directory 서비스 서버에 접속하여 등록된 이름을 사용해서 객체에 대한 객체 참조를 얻는다.

기업의 컴퓨터 네트워크 환경은 다른 여러 혼합된 Namespace를 나타내는 몇 개의 Naming기능으로 구성되는데 그 예로 전자 메일을 사용할 때 메일을 Namespace 다음에 보내고자하는 받는 사람의 이름을 정해주어야 한다. 각 객체들은 Naming 서비스 서버에 자신의 정보를 등록하고 클라이언트는 Naming 서비스 서버에 접속하여 등록된 이름

을 사용하여 객체에 대한 참조를 얻는다. Naming 서비스는 대부분 디렉토리 서비스로 확장되어지는데 Naming 서비스가 이름으로 주어진 객체를 조사하도록 제공해 주는데 반면 디렉토리 서비스는 그 개체의 속성(attributes)을 가지게 해준다. 그러므로 디렉토리 서비스는 조회하는 것 이외에 객체의 속성 혹은 주어진 속성의 객체(Object)까지도 조회할 수가 있다. Naming & Directory 서비스는 사용자, 기계, 네트워크, 서비스 응용프로그램 관한 다양한 정보를 널리 공유하여 인트라넷과 인터넷에서 중요한 역할을 한다.

Naming 서비스는 등록된 객체를 Fig. 4 같이 계층적인 트리 형태로 관리하게 된다. 이러한 트리의 구조로 관리되는 객체의 이름은 시스템 내부의 파일 시스템의 파일 이름처럼 계층적인 이름을 가지게 된다. 이러한 계층적인 트리 구조를 Naming 그래프라 한다.

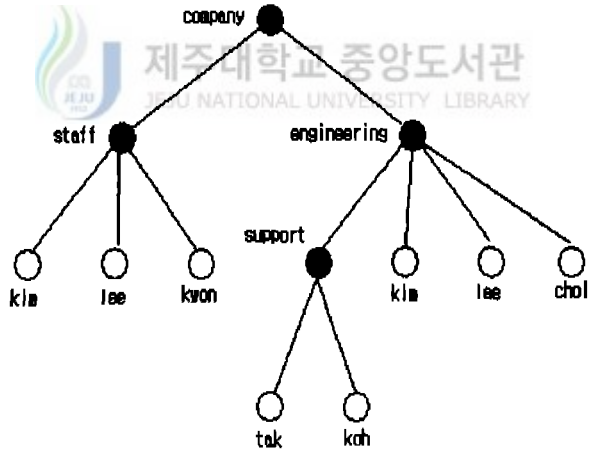

Fig. 4 Naming & Directory Graph

Fig. 4의 Naming 그래프에서 각각의 사원은 다음에서처럼 컨텍스트와 객체이름으로 구성된 자기만의 합성된 객체이름을 가지게 된다.

Fig. 4의 staff라는 부서에 일하는 kim라는 사원은 자신만의 독특한 합성이름인 company.staff.kim라는 합성이름(Compound Name)을 가지게 된다.

1) SNS(Simple Name Service)

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

2) DNS (Domain Name System)

사람이 인터넷의 규모와 호스트의 증가에 따른 IP주소만으로는 인식이 어려워 새로운 명명체계가 필요하여 등장하게 되었다. 특징으로는 계층적인 구조를 지니면, 대소문자를 구분하지 않으며, 분산환경에서 적용이 가능하며 클라이언트/서버에 구조를 갖는다. 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. 인터넷이 번창하는 한 디렉토리 서비스의 요구는 더욱 커질 것이다.(Douglas E. 등 1998)

Fig. 5는 DNS는 사용자 프로그램은 Local Name Server에 Query를 하면 Naming Sever의 프로세서는 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 Server로는 외부루트서버로서 먼저 사용자의 질의한 Name 를(www.netscape.com)를 외부루트서버(Root Name Server)에 질의를

한다. 그 외부루트서버가 그에 대한 IP를 Resolve하지 못하면 또 다른 외부Name서버(com Name Server)를 지역Name Server에게 지정하여 준다. 그러면 또다시 지역Name서버는 외부Name서버(com Name Server)에 또 다시 질의를 한다. 만약 외부Name Server(com Name Server)에 해당 Name이 없고 단지 해당 name을 갖고 있는 해당 지역 서버(netscape.com Name Server)를 지정하여준다. 이 해당 지역서버인 netscape.com에 최종 질의를 하여 www.netscape.com의 IP를 얻는다.

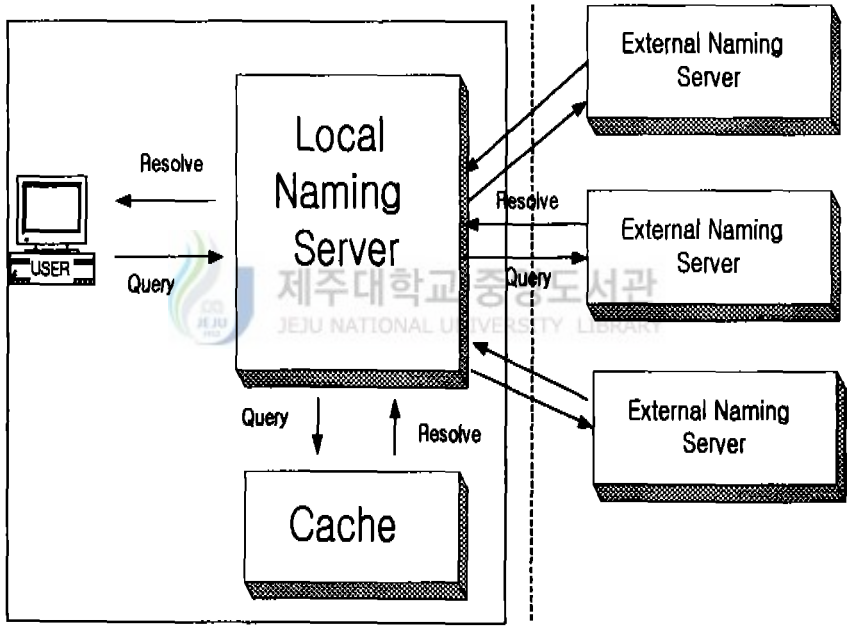

Fig. 5 Domain Name System Architecture

2. Directory Service

Directory 서비스는 사용자가 네트워크상에서 지능적으로 자원을 찾을 수 있는 애플리케이션을 구축할 수 있도록 한다. Directory는 애플리케이션을 대신해 위치를 파악하고 형태를 바꾸거나 움직이고 삭제되면서 자원들을 순회한다. 좋은 예로 전자우편의 애플리케이션은 유저그룹의 위치를 알아내고 워드프로세싱은 애플리케이션은 프린터를 찾아내며 클라이언트/서버 애플리케이션은 자원들 네트워크 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 서비스 서버에 등록되어 있다. 현재의 네트워크 Directory 서비스에는 DNS, 노벨의 네트웨어 Directory 시스템과 노벨 Directory 서비스, 넷스케이프의 Directory 서버, 마이크로소프트의 액티브 Directory와 CCITT의 X.500 등이 있다. (데이자베이스 월드 inc. 1999)

1) X.500

1988년 국제전신전화자문위원회(CCITT)는 X.500 directory 서비스를 출간하였다. X.500은 세계적인 디렉토리 정보의 보고로 유저 이름과 패스워드, 사진, 위치, 엑세스 컨트롤 메카니즘 등 귀중한 자료를 보유하고 있다. 애플리케이션이 자원을 쉽게 찾는 메카니즘을 제공하는 X.500은 DNS보다 적용하기 쉽고 기능과 특징이 더욱 다양하다는 장점을 갖는다. X.500 Directory 모델은 정보의 논리적 데이터베이스 제공을 돕는 시스템 중 하나로 분산된 집단 시스템이다. Directory 서비스 사용자는 이 정보를 이용하여 Directory 가입자에게 Directory 서비스를 엑세스 할 수 있다.

Directory 서비스는 Namespace에 의존하고 있는데 Namespace는 관련정보의 집단화를 효과적으로 참조, 검색할 수 있도록 한다. X.500 Namespace에서는 논리적이고 체계적인 디렉토리가 제공되며 엔트리의 구조를 다루고 있지만 유저에게 정보를 프리젠테이션하지는 않는다. X.500 DIT(Directory Information Tree)에 있는 모든 엔트리는 진정한 속성들의 모음이다. X.500은 국가, 지역, 조직, 조직단위, 사람, 별명 등의 객체 클래스를 정의한다.(William Stallings, 1996)

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. LADP은 응용 프로그램이 디렉토리 정보를 요청하고 관리하는 데 있어 표준적인 방법을 제공한다.

LDAP 프롤토콜은 X.500 DAP(Directory Access Protocol)로의 자원요청없이 X.500 모델들을 지원하는 디렉토리 액세스를 제공하기 위해서 제작되었다. 이 프로토콜은 특별히 디렉토리들 상호간의 읽기/쓰기 액세스를 제공하는 처리응용(Management Application)들과 브라우저 응용들을 목표로 삼았다. 이것은 X.500을 위한 보완물로서 사용되어진다. 세션계층과 프리젠테이션 계층을 우회하여 TCP나 다른 전송계층 프로토콜의 지렛대 역할을 수행 할 수 있다. X.500의 디렉토리 모델 엔트리에 기초하고 있으며 엔트리에 언급된 이름과 구별되는 이름을 사용하고 있다. 반면 고도로 구조화된 x.500 데이터 인코딩 메카니즘을 사용하는 대신에 LDAP는 엔트리를 표시하기 위해 단일한 문자열에 기초한 접근 방식을 시도하고 있다. 따라서 에플리케이션의 Directory 서

비스 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 LDAP을 위해 디자인되었다. 넷스케이프는 넷스케이프 커뮤니케이터에서 LDAP를 지원하고 있다. 대부분의 디렉토리들은 각각 몇몇 특정 디렉토리 작업을 위해 최적화되어 있다. X.500은 DAP을 위해 최적화 되어 있고, 노벨l Directory Server는 디렉토리 지원 NetWare 응용 프로그램을 위해 최적화 되어 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 Netscape Directory 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 메카니즘을 제공한다.

3. JNDI

우리는 현재 모든것을 자바화 하고 있다. 하물며 Directory 서비스라고 예외일 수 없다. 썬마이크로 시스템사의 자바소프트에서는 현재 JNDI로 알려져 있는 표준 자바 Directory상에서 작업을 하고 있다.

JNDI는 Naming & Directory 서비스를 제공하는 순수 Java API이다. JNDI를 사용하므로서 자바 애플리케이션은 물론 자바 애플릿을 통해 자바 객체를 저장하고 검색할 수 있다. 게다가 JNDI은 어트리뷰트를 사용하는 객체를 검색하거나 객체에 그 어트리뷰트를 연결하는것 같은 기본적인 오퍼레이션을 수행하기 위한 방법을 제공한다. JNDI는 Naming & Directory 서비스 구현에 독립적이다. 이것은 공통된 API을 사용하므로서 각양각색의 Naming & Directory 서비스를 자바응용프로그램을 통해 액세스할 수 있도록 하여준다. 즉, 표준 Directory 서비스를 수행할 수 있는 방법을 제공해 주고 있다.

여러 Naming & Directory 제공자는 공통된 API 뒤에서 단락없이 접속될 수 있다. 이것은 Java 응용프로그램이 LDAP(Lightweigh Directory Access Protocol), NDS(Naming and Directory Service), NIS(Network Information Service), DNS(Domain Name System) 그리고 CosNaming(CORBA Naming)같이 존재하는 다양한 Naming & Directory 서비스를 이용할 수 있도록 해준다. JNDI을 사용하므로서 자바 응용프로그래머는 강력하고 이식성이 강한 응용프로그램을 만들 수 있다. 이 강한 응용프로그램은 환경과 잘 접합될 수 있을 뿐만 아니라 자바의 객체 모델의 이점을 이용 할 수 있다.

Fig. 6을 보면은 JAVA Application에서 Naming & Directory 서비스를 받는다. 이 단계의 프로그램은 Java 언어로 만들어졌으며, 각 사용자의 필요에 따라 객체를 등록, 삭제, 조회, 질의 등의 작업이 여기에

서 이루어진다. JNDI API는 인터페이스로서 사용자의 응용프로그램이 Naming & Directory Service를 받을 수 있도록, Naming & Directory Server에 접속을 하여 오퍼레이션을 하는데 인터페이스를 제공한다. 이 인터페이스를 제공하는 Class Package는 뒤에 설명이 있다. JNDI Namig Manager는 서비스를 제공하기 위해서는 Server가 존재해야 하는데 이 서버를 의미한다. JNDI SPI는 각 Naming & Directory를 제공하는 서비스 Provider, 즉 여러 서비스 제공환경(LDAP, DNS, NIS, RMI, CORBA 등)이 이루어지도록 인터페이스를 제공한다.

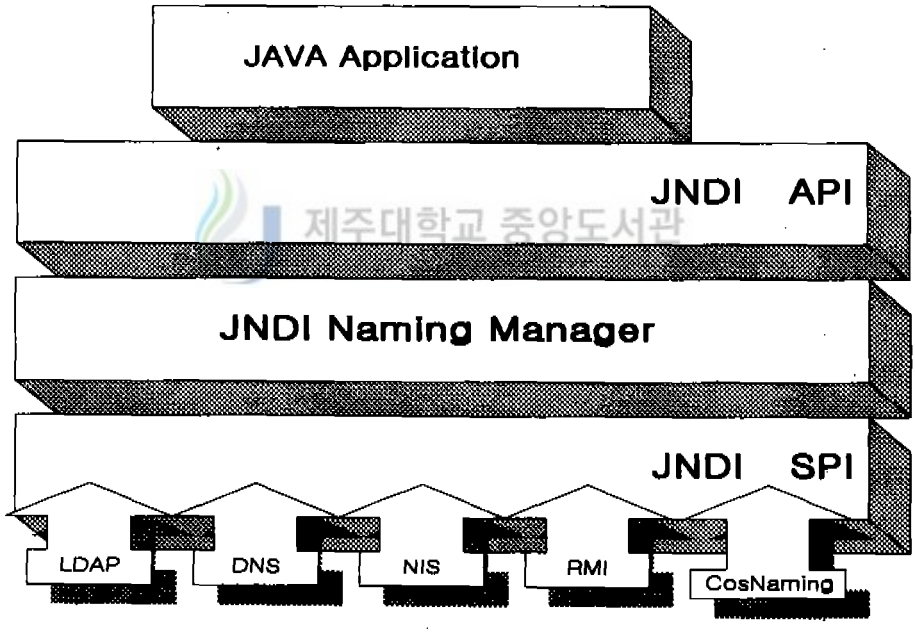

Fig. 6 The architecture of JNDI

JNDI은 다음의 인터페이스 package class로 구성되어졌다.

- javax.naming.Directory
- javax.naming.Directory
- javax.naming.event
- javax.naming.ldap
- javax.naming.spi

각각의 package class를 보면

1) javax.naming.directory

Naming & Directory 문장을 규정하는 가장 핵심되는 context이다. 이것은 객체와 이름을 묶어 추가하는 것(adding), 객체를 조회(looking up)것, 객체를 열거하는 것(listing), 등록된 객체를 제거하는 것(removing), 같은 타입의 서브 context의 생성 및 제거(creating and destroying)같은 기본적인 동작을 정의한다. context.lookup( ) 은 가장 공통적으로 사용되는 오퍼레이션이다. 이 context 구현은 자바응용프로그램에 의해 요구되어 질 때마다 객체를 돌려줄 수 있다. 예를 들면, 응용프로그램이 일치하는 프린터 객체를 조회하고, 직접 프린트작업을 하기 위해 프린트라는 이름를 응용프로그램 내에 사용한다.

```
Printer printer = (Printer)
cts.lookup("treekiller");
printer.print(report);
```

이 응용프로그램은 Naming & Directory 서비스를 받기 위해 어떠한 구현을 한 것이 없다. 즉, JNDI 에서 인터페이스을 제공해주기 때문에 가능한 것이다.

2) javax.naming.directory

DirContext 인터페이스는 디렉토리 객체와 연관되여진 어트리뷰트을 갱신하고 시험하기 위하여 메서드를 정의하므로서 디렉토리 서비스를 가능하게 해준다. 각 디렉토리 객체는 제로 또는 그이상의 어트리뷰트 class을 갖는다. 각 어트리뷰트는 문자 구분자에 의해 구분 되여진다. 그리고 어떤 형태든지 값을 가질수 있다. 디렉토리 객체는 다양한 정보를 보유한다. 예를 들어 한사람과 연관되여진 자원에 대한 Natural Naming & Directory Context가 있다. 즉, 그 사람의 프린터, 그 사람의 파일시스템 등이 Natural Naming & Directory context가 되는것이다. DirContext.Secarch()는 디렉토리 객체에 대한 메서드이다. 이것은 응용프로그램이 명확한 어트리뷰트의 세트를 값을 주면 요구 되여진 어트리뷰트 값과 함께 대응하는 디렉토리 객체를 알려준다.

3) java.naming.event

이 클래스는 Naming & Directory 서비스에 의해 발생하는 이벤트를 나타낸다. NamingEvent의 예로는 디렉토리 내용의 재명명 혹은 디렉토리 내용의 어트리뷰트 값의 변경이 있다. NamingListener는 NamingEvent에서 발생하는 사항을 기록하는 객체이다. Listener는 context, context의 children 혹은 서브트리 안에서 변화하여 발생하는 알림을 받는 context를 가지고 기록한다.

4) javax.naming.ldap

LdapContext는 응용프로그램에게 LDAP v3을 사용 할 수 있게 인터페이스를 제공한다.

5) java.naming.spi

이것은 여러 Naming & Directory 서비스 provider에 의해 만들어진 것 프로그램에 의하여 JNDI를 사용하는 응용프로그램이 대응하는 서비

스를 액세스 할 수 있도록 수단을 제공한다. 또 JNDI는 다른 다중 Namespace의 확장이 가능하므로 한 개의 서비스 Provider 는 오퍼레이션을 완성하기 위해 다른 서비스 Provider 와 상호작용을 할 필요가 있다. 이때 SPI는 한 개의 서비스 Provider가 다른 서비스 Provider에게 상호작용을 할 수 있도록 메서드를 제공한다. (Sun Microsystems, inc., 1999)

4. JNDI를 이용한 Mobile Agent 네트워크관리

Mobile Agent 기반 네트워크관리는 앞 절에서 언급한 것과 같이 관리자가 자신의 역할을 Mobile Agent에 일임하여, 직접 SNMP대리자가 있는 대상시스템으로 Mobile Agent를 파견하게된다. Mobile Agent는 맡겨진 임무를 수행하여 관리자의 역할을 대신하여 SNMP대리자와 오퍼레이션을 하여 MIB에 있는 SNMP대리자로 하여금 관리정보를 가져오게 하여 그 오퍼레이션 동작결과를 관리시스템으로 보내게 되는 구조를 가진다. 이처럼 관리자가 Mobile Agent를 생성하고 파견하는 것에 대해 Mobile Agent를 이용한 기존의 네트워크관리들은 관리자가 SNMP대리자를 선택하는 부분에 있어 어떠한 참조 또는 정보를 제공받지 못하였다. 그래서 Mobile Agent가 분산환경에서 이동해야 할 곳을 신속하고 정확하게 위치파악을 하지 못하여 네트워크관리하는 관리시스템에게는 분산네트워크관리 기능을 제대로 수행하지 못하는 결과가 될 것이다.

본 논문에서 제안한 구조에서는 관리시스템은 Naming & Directory 서버로부터 객체들을 참조할 수 있기 때문에 분산환경에서 중요하게 여겨지는 네트워크 투명성 중 위치투명성을 제공받을 수 있어 Mobile Agent기반 분산 네트워크관리에 무척 효율적으로 사용될 수 있다.

JNDI는 Naming & Directory 서비스를 하기 위한 인터페이스를 제공하는 API(Application Program Interface)으로서 여러 Naming 및 Diretory 기능을 수행하는 여러 서비스 가능 환경(서비스 Provider : CORBA, DNS, NIS, LDAP 등)을 동시에 연동 가능하게 하여 Mobile Agent가 여러 가지의 환경에서 서비스를 받아 추가 기능을 Mobile Agent에 더할 수 있는 것이다.

Ⅳ. 시스템 설계 및 구현

Naming & Directory 서비스를 Mobile Agent 기반 네트워크관리시스템에 적용하기 위하여 제안한 구조와 구현 결과는 다음과 같다.

1. 시스템 구조

Fig. 7은 본 논문에서 제안하는 Mobile Agent 기반 네트워크관리시스템의 전체적인 구조를 나타내고 있다.

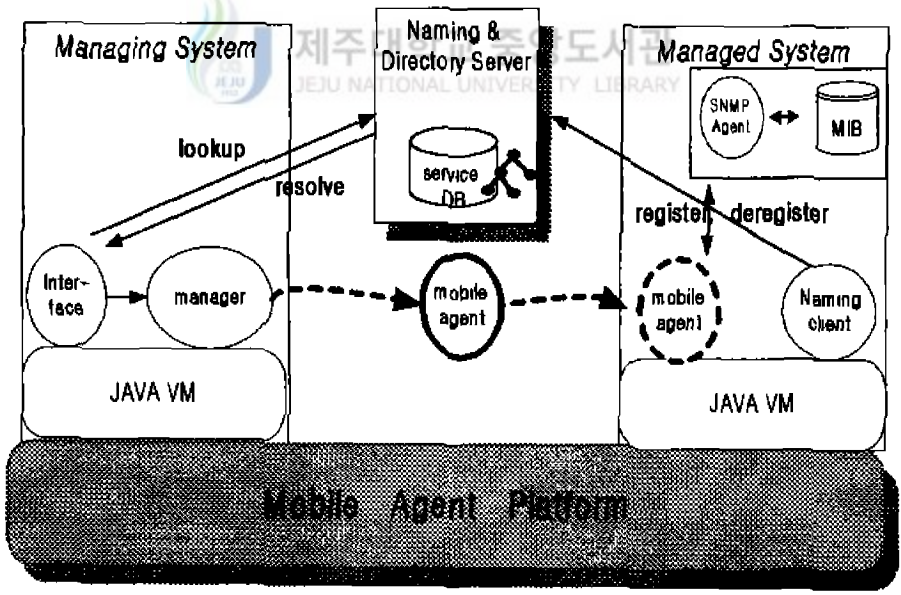

Fig. 7 The Architecture of Supposed System.

Fig. 7은 네트워크관리를 하기 위해 관리자는 Naming & Directory 서버(Netscape Directory Sever 4.1)의 DB로부터 이동 할 관리대상시스템의 정보를 참조 받아 해당 관리대상시스템에 Mobile Agent를 파견시킬 수 있다. Mobile Agent는 Proxy를 통해 관리대상시스템의 SNMP대리자와 SNMP 오퍼레이션을 수행하여 SNMP대리자는 MIB(Managemet Inforamtion Base)의 관리정보를 찾아 결과를 다시 Mobile Agent에게 넘기면 Mobile Agent는 그 결과 값을 관리시스템에게 전송한다.

2. 시스템 구현 환경

구현된 시스템의 환경은 다음과 같다.

- Managing system의 환경.
  - 운영체제 : 솔라리스 2.6.
  - 시스템사양 : SUN 워크스테이션(SUN Ultra-2).
- Managed system의 환경.
  - 운영체제 : 솔라리스 7.
  - 시스템사양 : X86계열 Pentium Processor.
- Naming & Directory Server의 환경.
  - 운영체제 : Window NT.
  - 시스템사양 : X86계열 PentiumⅡ, Dual Processor.
- 시스템 구현에 사용된 언어(도구)
  - JDK v1.2
  - JAVA Beans(AdventNet SNMPv1 API class)
  - Voyager ORB 3.1.2

- JNDI 1.2
- Netscape LDAP Directory Server 4.1

3. Global Naming Tree

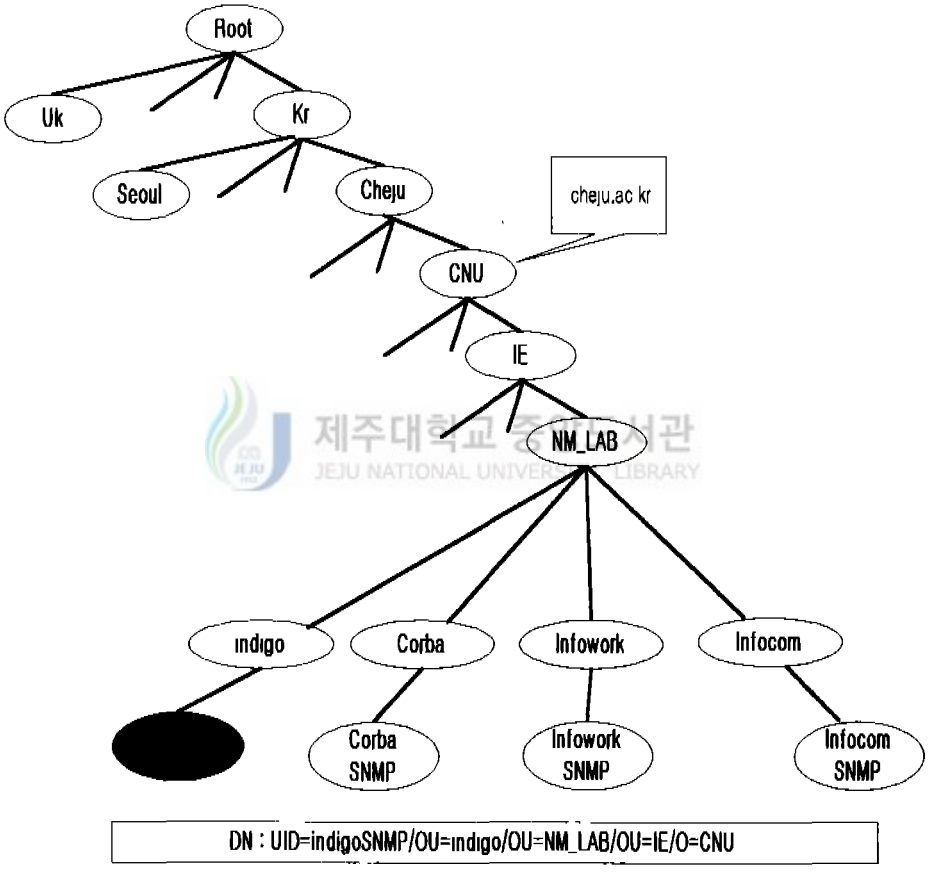

Fig. 8 Global Naming Tree

본 논문에서는 LDAP Directory Server(Netscape Directory Server4.1)를 통해 Namig Service를 하기 위해 객체를 Fig. 8 같은 트

리구조로 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, InfocomSNMP라는 이름으로 등록시켰다. 본 구현 시스템에서는 CNU를 Directory Root로 하여 설정하였기 때문에 DN(Distinguish Name)의 O=cheju.ac.kr를 루트로 등록시켰다.

4. Naming & Directory Service 스키마 구성

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

- Register : Naming & Directory Server에 객체를 등록시키기 위해 다음과 같은 인수 값을 입력하였다.

```
ADD([in]Name DN,
    [in]SuperName SuperDN,
    [in]MyType ObjectType
    [in]Host HostInfo);
```

- Deregister : 객체를 삭제시키는 위한 입력값

```
Delete([in]Name DN);
```

· Search : 객체에 대한 속성값으로 조회

```
Search([in]MyType ObjectType,
       [[out]Host HostInfo]*);
```

· LookUp : 객체의 이름으로 조회

```
LookUp([in]Name RDN,
       [out]MyType ObjectType.
       [out]Host HostInfo);
```

5. 시스템 동작과정 및 결과

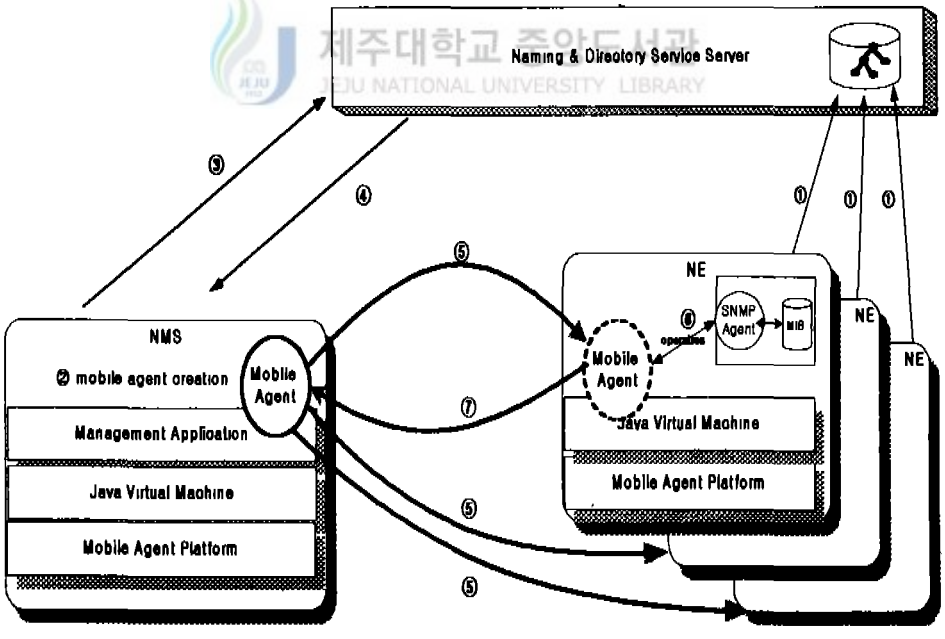

Fig. 9 The System Procedure

Fig. 9와 같이 구현시스템은 다음의 순서로 동작한다.

① 관리대상시스템이 JNDI서버에게 자신의 객체참조를 등록시킨다.

② 관리시스템의 응용프로그램은 Mobile Agent를 생성한다.

③ JNDI Server에 찾고자 하는 이름의 객체참조를 조회한다.

④ 등록된 객체참조를 받아 Mobile Agent에 그 참조값을 넘겨준다.

⑤ Mobile Agent 는 관리대상시스템으로 이동한다.

⑥ Mobile Agent에 부여된 의무는 SNMP대리자와 오퍼레이션을 한다.

⑦ SNMP 대리자는 MIB-II값을 가져와 Mobile Agent에 넘겨 Mobile Agent는 다시 그 값을 관리시스템으로 넘긴다.

앞의 과정중 ①번에 해당하는 등록과정을 Fig. 10에서 보여주고 있다.

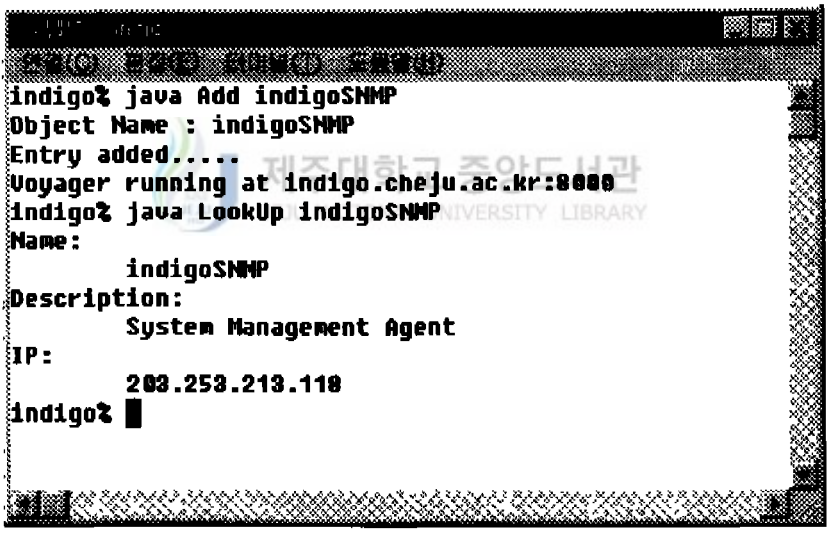

Fig. 10 Registering and Lookuping NE on Directory Server

Fig. 10은 Directory 서버에 대상시스템(indigo)의 SNMP대리자를 "Corba" 시스템(NT, Directory server가 설치되었음.)에서 텔넷으로 indigo에 있는 JNDI를 이용한 애플리케이션(디렉토리 클라이언트 프로

그램)을 실행시킨 장면이다. 등록정보는 Name, Description, IP로 정하여 Directory Server에 등록시키는 과정을 간단히 보여주고 있다.

등록시킬 때 object의 위치(DN: Distinguish Name)를 지정해 주어야 한다.

DN:UID=indigoSNMP, OU=indigo, OU=NM_LAB, OU=IE, O=cheju.ac.kr

이 indigoSNMP CN(Common Name)를 입력하기 위해서는 사전에 indigo라는 OU(Organization Unit)까지 미리 설정 되어있어야 추가가 가능하다.

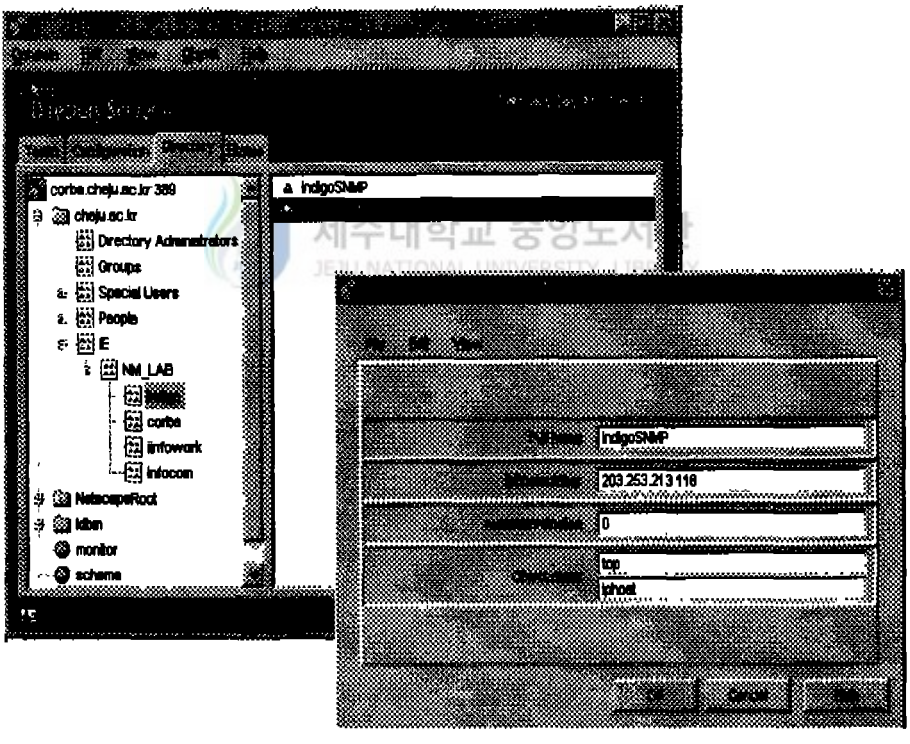

Fig. 11 Display of Naming & Directory Server

Fig.11은 각 관리대상시스템을 Naming & Directory Server에 등록시

킨 장면을 잡은 화면이다. 그 중 indigo대상시스템의 indigoSNMP에 Mobile Agent를 보내 동작한 결과를 관리시스템(Infowork)에서 보여주는 화면이다.

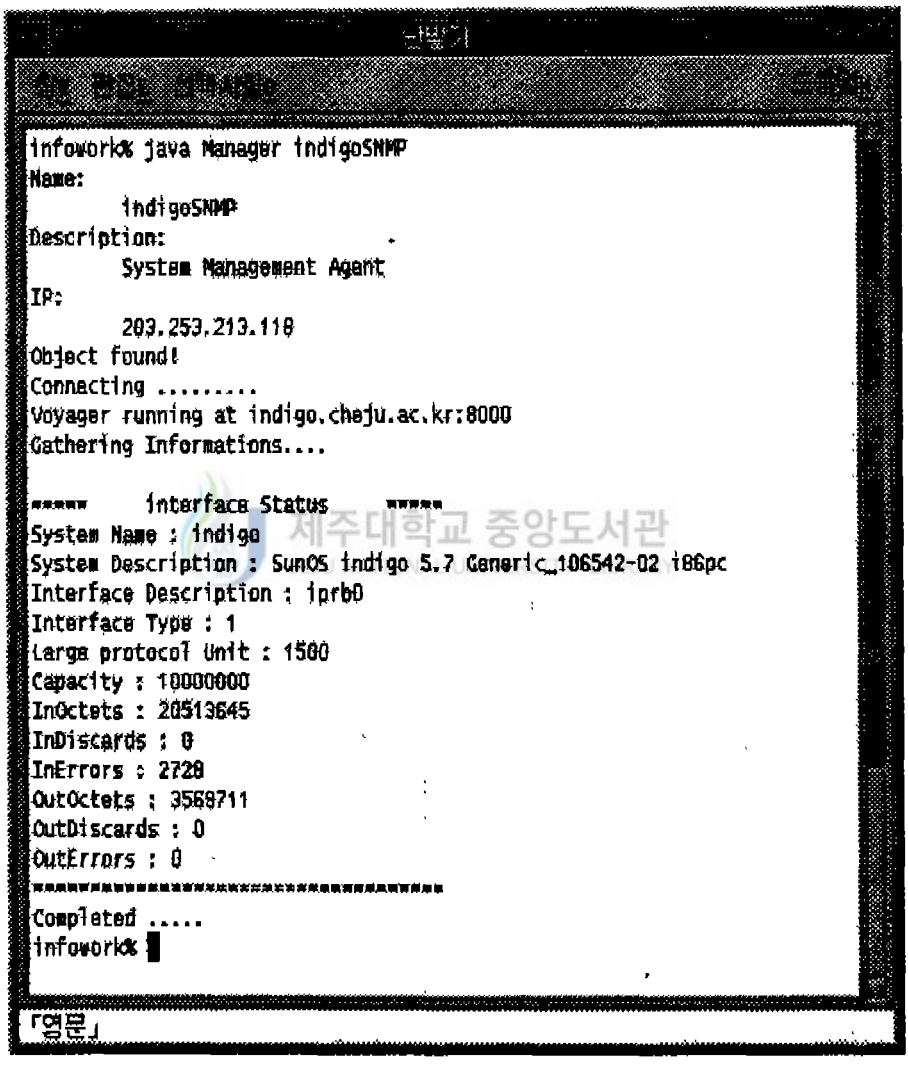

Fig. 12 Display of Managing System

Fig. 12는 관리자가 indigoSNMP라는 관리대상시스템을 관리하기 위해 Naming & Directory Server에게 질의를 하여 해당 Mobile Agent가 이동할 수 있는 관리대상시스템의 IP값을 인수로 받는다. 관리대상시스템에는 Voyager ORB 플랫폼이 형성되어 있음을 directory service를 통해서 확인할 수 있다. IP값을 받은 Mobile Agent가 그 관리대상시스템으로 이동하여 시스템 인터페이스정보(System Name, System 일반정보)와 시스템의 Network 인터페이스정보(LAN 카드의 일반정보, LAN 카드의 종류, LAN을 통한 카드의 입력 패킷 수, LAN을 통한 출력 패킷의 수, LAN를 통한 버려지는 패킷 수, LAN을 통한 에러가 발생한 패킷 수)를 수집하여 그 값을 관리시스템으로 보낸다.

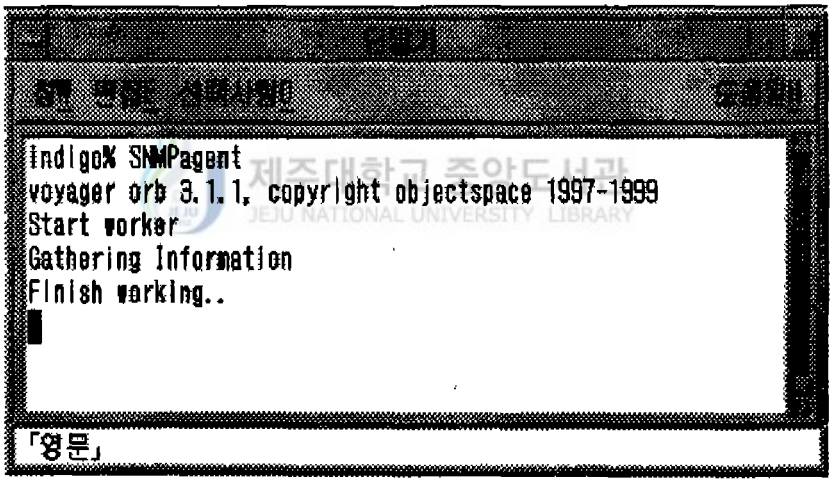

Fig. 13 Display of Managed System

Fig. 13은 관리대상시스템의 동작수행결과이다. Mobile Agent가 관리대상시스템에 도달하여 SNMP대리자와 동작하여 정보를 수집하고 있는 과정이다.

V. 결 론

그 동안 소규모 사업자의 네트워크그룹(중앙집중식 네트워크)에서는 애플리케이션에는 글로벌 네트워크 자원을 검토하고 액세스할 순차적인 메카니즘이 필요치 않았다. 하지만 지금은 이러한 소규모 애플리케이션은 설 자리를 잃어가고 있다. 현재, 빌딩, 도시, 지방, 국가에 이르기까지 모두 기업수준의 애플리케이션을 구축하고 있다. 따라서 Naming & Directory 서비스는 수 백가지 네트워크 상에서 액세스할 수 있는 자원을 이용하여 수 천명의 사용자를 지원하게 된다. 애플리케이션을 구축하고 자원을 사용하는 방법은 간단하지 않지만 이 네트워크 관리자라면 서비스를 이해하고 이용할 줄 알아야 한다.

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

그 결과로 관리자는 대리자에 대한 세밀한 정보 혹은 네트워크 특성과 위치 특성에 따른 부분들을 고려하지 않아도 됨에 따라 Mobile Agent 기반의 네트워크 관리에 네트워크 투명성을 제공할 수 있어 이것은 분산환경에서 무척이나 중요하게 점으로 작용할 수 있었다.

향후 연구 과제로는 Directory Server부터 추가적인 기능을 받기 위해Directory 서비스 중 객체를 참조하는데 있어서 이름뿐만 아니라 객체의 속성값(Attribute value)을 가지고 객체의 정보를 얻을 수 있는

메카니즘이이루어질 수 있도록 데이터베이스의 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 JNDI을 사용한 Naming & Directory 서비스를 이용한 Mobile Agent 기반 네트워크 모델은 네트워크 관리자, 관리시스템, 관리대상시스템 모두에게 여러가지 면에서 효율적이기 때문에 앞으로 계속적인 연구와 개발을 통하여 훌륭한 네트워크관리 모델이 될 것으로 고려된다.

참고 문헌

Advent Network Management inc., 1999, "Advent SNMP Package" "http://www.adventnet.com/java-nm-resources.html"

조수형, 1999, "CORBA 기반에서 SNMP와의 연도에 관한 설계 및 구현", 석사학위논문, 전북대학교대학원 영상정보공학과, pp.12-13.

조일권, 1997, "Mobile Agent를 이용한 망관리방법에 관한 연구", 석사학위논문, 한양대학교 대학원, 전자공학과, pp.16-17.

정보통신부, 1993, "단순망 관리 규약(SNMP) 표준" pp.8-37.

George Coulouris, Jean Dollimore Tim Kindberg, 1997, "Distributed Systems Concepts and Design Second Edition" Addison-Wesley, pp.643.

마복순 등, 1999, "네트워크보안관리를 위한 Mobile Agent 설계" 홍익대학교. 한국정보통신학회 논문.

데이터베이스 월드 inc., 1998, "디렉토리서비스 향배" 58호 pp.87-91.

Douglas E. Comer, David L. Stevens, 1998, "Internetworking with TCP/IP volumeIII Client-Server Programming And Applications", Prentice Hall.

왕창종 · 이세훈, 1999, "Inside CORBA3 프로그래밍" 도서출판대림, pp.285~316.

한국전산원, 1995, "통합망관리 표준화 연구" pp.114~119.

김정철 · 송왕철, 1999. "JNDI를 이용한 Mobile Agent 기반 망관리시스템", 한국해양정보통신학회추계종합학술발표대회, pp.406~410.

Weiyi(william) Li, David G. Messerschmitt, 1999, "Java-to-Go", California at Berkeley.

Danny B. Lange, Mitsuru Oshima, 1998, "Programming and Deploying Java Mobile Agents with Aglets", Addison-Wesley, pp.1~30.

이인화, 1994, "Corporate LAN상에서 최적의 네트워크 관리 구현 방안", 석사학위논문, 중앙대학교 정보산업대학원, 컴퓨터 소프트웨어학과, pp.4-6.

Netscape inc., 1999, "Netscape Directory Server version4.1 Installation Guide", "Server Administrator Guide", "Deployment Guide", "Managing Server with Console".

ObjectSpace inc., 1999, "Voyager ORB3.0 Developer Guide", pp.309 Object Management Group December, 1997. "The Common Object Request Broker: Architecture and Specification".

David Perkins, Evan McGinnis; 1997, "Understanding SNMP MIBs" Perntice Hall, pp.15~30, pp.120~150.

Vu Anh Phamn and Ahmed Karmouch, 1998, "Mobile Software Agent Overview". IEEE communication Magazine, pp.26~29.

Morris Sloman, 1996, "Network and Distributed System Management", Addison-Wesley, pp.250~280.

William Stallings, 1996, "SNMP SNMPv2 and RMON", Addison-Wesley Publishing Company, Addison-Wesley, pp.84~120, pp.145~196.

Sun Microsystems, inc., 1999, "Java Naming Directory Interface Application Program Interface(JNDI, API, SPI)", pp.68, pp.48.

감사의 글

제가 이순간이 있기까지 도와주신 당신께 이 논문을 바칩니다.

지도교수로서 지도와 편달을 아끼지 않은 송왕철 고수님께 먼저 감사드립니다. 논문의 완성되기까지 작은 흠까지도 지적해주신 김장형 고수님, 안기종 고수님, 곽호영 고수님, 변상용 고수님, 이상준 고수님, 이동희 고수님 감사합니다.

직장에 다니면서 제대로 논문을 쓸 수 있을까 의심도 많이 해보았고 좌절도 했습니다. 전 지금까지 이처럼 많은 시간과 노력을 들려 공부해 본적이 없습니다. 이런 저에게 포기하지 않게 격정해주고 격려해주신 부모님에게 이 논문을 바치겠습니다. 그리고 장인 장모님에게도 이 논문을 바치고 싶습니다. 공부한답시고 집안 대소사에 거의 빠져도 이해해주신 형님, 형수님 가족 모두에게 감사합니다. 적당히 눈감아 주신 직장상사님, 많은 자료와 지원을 해준 대학원 선후배 동료, 그동안 소원해진 친구 모두에게 고맙다는 말씀으로 대신하겠습니다.

마지막으로, 뭐라고 표현해야 적당할지, 너무나 착하고 이쁜 아내에게 이 논문을 바칩니다. 대학원 다니는 동안 막내 "가희"가 태어나 직장이 있는 아내의 고생은 몇 배는 더했을 것입니다. 그리고 아이들에게 남기고 싶은 말은 "얘들아! 아빠는 힘들 때 항상 너희들을 생각했단다."