그룹통신을 이용한 견고한 LDAP 서버

문 남 두[†] · 안 건 태[†] · 박 양 수[††] · 이 명 준[††]

요 약

LDAP(Lightweight Directory Access Protocol) 디렉토리 서비스는 인터넷이나 인트라넷 등 네트워크 상에 있는 파일이나 장치들과 같은 자원의 위치를 찾을 수 있도록 정보를 제공한다. LDAP는 인터넷에서 표준 디렉토리 서비스 구조로 폭넓게 받아들여지고 있으므로, 다수의 LDAP 서버들 사이에 디렉토리 정보를 중복하여 유지함으로써 특정 서버와의 네트워크 단절(partition)과 같은 결함이 발생되는 상황에서도 투명하고 지속적으로 서비스를 제공하는 것이 바람직하다. 본 논문에서는 *JACE 그룹통신 시스템의 프로세스 그룹*으로 동작하는 견고한 *LDAP 서버 시스템*과 자바 응용 프로그램에서 서비스를 사용할 수 있도록 그룹통신을 이용하는 *LDAP 서비스 제공자*의 설계와 구현에 대하여 기술한다.

A Robust LDAP Server Using Group Communication

Nam-Doo Moon[†] · Geon-Tae Ahn[†] · Yang-Soo Park[††] · Myung-Joon Lee[††]

ABSTRACT

LDAP (Lightweight Directory Access Protocol) Directory Service provides information for locating resources like files and devices over the network such as Internet or Intranet. Since LDAP is widely accepted as one of the standard directory service structure for the Internet, it is desirable that a group of LDAP servers works transparently and continuously even if the related network partitions temporally, through maintaining replicated directory information among those LDAP servers. In this paper, we describe the design and implementation of a *robust LDAP server*, which runs as a *process group* in *JACE group communication system*, and the associated *LDAP service provider* which enables Java applications to use the developed LDAP directory service.

키워드 : LDAP, 디렉토리 서비스(Directory Service), JACE 그룹통신 시스템(JACE Group Communication System), LDAP 서비스 제공자(LDAP Service Provider)

1. 서 론

최근의 인터넷이나 기업 내의 인트라넷 등 네트워크 환경은 사용자 PC, 프린터, 응용 서비스, 네트워크 장치, 파일, 자바 객체 등 다양한 자원(resources)이 사용될 수 있게 변화되고 있다. 이와 같은 환경에서, 디렉토리(Directory) 서비스는 다양한 색인, 캐싱, 디스크 접근 기술들을 통하여 네트워크의 다양한 자원과 관련된 정보를 논리적으로 접근할 수 있도록 지원한다. 디렉토리 서비스는 정보를 빠르게 읽을 수 있도록 고안된 특별한 형식의 데이터베이스이다. 기존의 디렉토리 서비스에는 NDS(Novell Directory Services), LDAP(Lightweight Directory Access Protocol)[1, 2], NIS(Network Information System) 등이 있다. LDAP는 X.500 프로토콜을 모태로 하고 있으며, 일반적이고 산업계에서 폭넓게 사용되고 있다. LDAP는 클라이언트-서버 기반의 디렉토리 서비스를 제공해 준다. 기본적인 기능은 기업의 정보 등을 포함한 다양한 정보를 저장할 수 있으며 이러한 정보를 검색, 변경, 삭제할 수 있다. 인터넷 표준으로서 LDAP의 출현은 디렉토리 상호 운용성과 디렉토리 가능 응용 프로그램 개발을 가능하게 하기 때문에 중요하다. 따라서, 분산된 디렉토리 서버들 사이의 디렉토리 정보를 중복하여 유지함으로써 네트워크 단절(partition), 특정 서버의 실패(fail)과 같은 오류가 발생하더라도 투명하고 지속적인 서비스를 제공하는 것이 바람직하다.

JACE(Java Advanced Communication Environments)[3-5] 그룹통신 시스템은 동일한 서비스를 제공하는 다수의 응용 서비스가 프로세스 그룹으로 동작될 수 있도록 지원한다. JACE 기반의 응용 프로세스 그룹에 참여하는 응용 서비스의 상태는 일시적인 네트워크 단절이나 서버의 실패가 발생하는 상황에서도 동일하게 유지된다. 사용자는 응용 서버의 특정 위치에 종속되지 않고 이용 가능한 서버로부터 서비스를 제공받게 된다.

본 논문에서는 그룹통신 기반의 프로세스 그룹으로 동작하는 LDAP 서버와 이를 이용하기 위한 LDAP 서비스 제공자의 설계와 구현에 관하여 기술한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로서 우선 자바 기술 중에서 네이밍과 디렉토리 서비스를 이용

※ 본 연구는 정보통신부 대학기초연구지원 사업과제의 지원으로 수행되었음.
† 준 회 원 : 울산대학교 대학원 컴퓨터정보통신 공학부
†† 정 회 원 : 울산대학교 컴퓨터정보통신 공학부 교수
논문접수 : 2002년 4월 3일, 심사완료 : 2003년 2월 19일

하기 위한 JNDI(Java Naming and Directory Interface)[6]를 살펴본다. 그리고 경량의 디렉토리 서비스를 제공하는 LDAP 디렉토리 서비스의 구조와 기능에 대하여 설명하고, 기존의 JNDI와 LDAP 서버와의 연관 관계 및 문제점을 제시한다. 또한 그룹통신 시스템의 전반적인 구조와 기능에 대하여 소개한다. 3장에서는 네트워크의 단절과 서버의 실패가 발생되는 환경에서도 투명하고 지속적인 디렉토리 서비스 제공을 위한 그룹통신 기반의 LDAP 서버와 서비스 제공자의 설계에 관하여 기술한다. 4장에서는 프로세스 그룹으로 동작하는 그룹통신 기반의 LDAP 서버와 이를 이용하기 위한 LDAP 서비스 제공자의 구현에 관하여 살펴보고, 개발된 시스템의 성능 평가에 관하여 기술한다. 끝으로 5장에서 결론 및 향후 연구방향에 대해 살펴본다.

2. 관련 연구

본 장에서는 JNDI와 LDAP 디렉토리 서비스에 대해 설명하고 이 둘간의 관계를 도식적으로 표현한다. 2.3절에서는 그룹통신 시스템 구조와 기능에 대해 설명한다. 또한 그룹통신 시스템을 이용한 신뢰성 있는 응용 프로그램 개발의 국내외 현황을 소개한다.

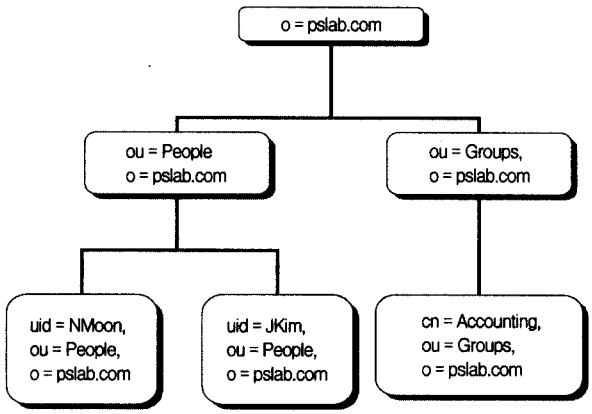
2.1 LDAP와 JNDI

JNDI는 자바 응용 프로그램들이 DNS, LDAP 및 NDS 등과 같은 네이밍 및 디렉토리 서비스를 이용하기 위한 썬의 API이다. 일반적으로 서로 다른 디렉토리 서비스는 각자의 서로 다른 API를 제공한다. JNDI는 서로 다른 디렉토리 서비스에 대하여 표준화된 API를 제공한다. 응용 프로그램 개발자들은 JNDI API만을 사용하여 다양한 디렉토리 서비스를 이용할 수 있다. 실질적으로 디렉토리 서비스와 상호 통신하는 디렉토리 드라이버는 JNDI SPI에 작성되어 있다. 새로운 디렉토리 서비스를 개발하였다면 개발된 디렉토리 서비스와 상호통신 할 수 있는 JNDI SPI를 개발함으로써, JNDI를 지원하게 된다.

2.1.1 LDAP 디렉토리 서비스

디렉토리는 검색할 정보의 커다란 집합으로 볼 수 있다. 전화번호부와 같이 정보에 거의 변화가 없고, 매우 자주 검색되는 정보는 디렉토리의 영역이라고 볼 수 있다. LDAP는 국제 표준기구(ISO)와 국제 원거리 통신 연맹(ITU)에서 승인한 디렉토리 표준인 X.500 기술의 문제점을 극복하기 위해 고안되었다. X.500은 보편적인 디렉토리를 만들어내고 그에 접속하는 데, 필요한 모든 프로토콜과 표준명세를 지원하고는 있지만 너무 크고 복잡하다. LDAP는 인터넷의 TCP/IP 프로토콜을 사용할 수 있도록 설계되었는데, X.500과 비교하면 자원소모가 적고 기능이 단순화되었다. 네트워크 디렉토리를 특수한 데이터베이스로 생각한다면 LDAP를 이용하여 관련 서버와 서버 주변기기, 어플리케이션이나 문서의 위치를 파악하는데도 많은 도움을 주므로 그 기능을 크게 확장시켜 나갈 수 있다.

(그림 1)은 LDAP 디렉토리 구조의 대표적인 예를 보여준다. 디렉토리 내의 정보는 보는 바와 같이 계층적으로 관리된다. 이러한 LDAP 디렉토리 구조를 특별히 DIT(Directory Information Tree)라고 부른다. LDAP 트리 구조에서 각 노드를 엔트리(Entry)라고 부르고, 엔트리는 LDAP에서 하나의 데이터를 나타낸다. 관계형 데이터베이스와 비교를 한다면 하나의 레코드와 일치한다고 할 수 있다. 모든 엔트리는 그 자신의 위치와 고유성을 나타내는 DN(Distinguished Name)으로 구분된다. 이는 파일 시스템과 유사하다. 예를 들어, (그림 1)에서 최하위 왼쪽 엔트리에 해당되는 DN은 uid = NMoon, ou = People, o = pslab.com이다.



(그림 1) 디렉토리 내의 데이터 조직도

LDAP 디렉토리 서버는 엔트리 검색, 엔트리 추가, 엔트리 변경, 그리고 엔트리 삭제 등의 주요 기능을 제공한다.

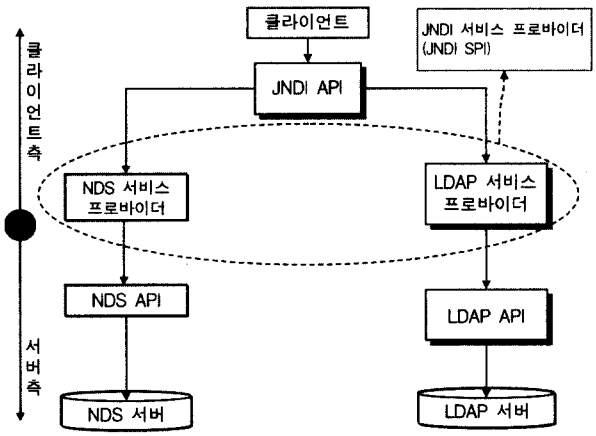
2.1.2 JNDI

서로 상이한 다수의 명명 서비스와 디렉토리 서비스가 기업 내에서 사용된다. 파일과 프린터 공유에 대해서는 NDS (Novel Directory Services)가 사용되고 전자우편이나 웹 페이지 공지를 위해서는 NIS(Network Information System)가 사용될 수 있다. JNDI 단일의 인터페이스를 사용하여 상이한 명명 서비스 및 디렉토리 서비스를 관리할 수 있도록 지원한다. JNDI는 <표 1>와 같은 패키지로 구성되어 있다.

〈표 1〉 JNDI 패키지

패 키 지	설 명
javax.naming	응용 프로그램이 명명 서비스와 대화하기 위한 API를 제공한다.
javax.naming.directory	응용 프로그램이 디렉토리 서비스와 대화하기 위한 API를 제공한다.
javax.naming.spi	JNDI API가 기존의 명명 서비스와 디렉토리 서비스와 상호 통신할 수 있게 지원한다.
javax.naming.ldap	응용 프로그램이 LDAP v3의 확장된 동작과 컨트롤을 사용할 수 있게 지원한다.

네이밍 및 디렉토리 서비스 개발자들은 응용 프로그램과 네이밍 및 디렉토리 서버가 상호통신할 수 있는 JNDI SPI [7,8]제공함으로써, 일반 JNDI 응용 프로그램 개발자들이 해당 서비스를 이용할 수 있도록 지원한다. JNDI와 LDAP 디렉토리 서비스의 기존 모델은 (그림 2)에서 오른쪽 흐름과 같이 클라이언트 측의 JNDI 호출이 내부적으로 LDAP 서비스 제공자 클래스에 의해 그 요청이 직접적으로 LDAP 서버에 전달되고, 연결되어 상호 통신한다. 이러한 모델의 사용은 오늘날과 같이 일시적인 네트워크 단절과 호스트의 실패(crash)가 발생되는 환경에서 가용성 높은 서비스를 제공하기 어렵다.



(그림 2) JNDI와 LDAP 디렉토리 서비스의 기존 모델

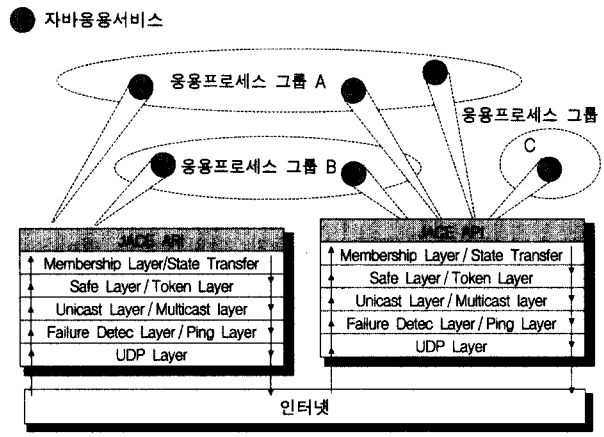
2.2 그룹통신 시스템

이 절에서는, 동일한 서비스를 제공하는 응용 서버들이 *프로세스 그룹*의 형태로 동작될 수 있도록 지원하는 JACE 그룹통신 시스템의 전반적인 구조와 기능에 대하여 기술한다. 국외에서 개발된 그룹통신 시스템으로는 Horus, Ensemble, Transis[9], Totem[10], JavaGroup[11], Jgroup[12] 등이 있다. JACE 시스템은 Totem 시스템과 같이 그룹에 참여하는 프로세스 멤버들 사이에 가상 링을 형성하여 동작한다. 또한 (그림 3)에서 보는바와 같이 시스템 구조를 기능별로 계층화함으로써 시스템 수정과 확장이 용이하도록 하였다.

2.2.1 JACE 그룹통신 시스템

인터넷 인구의 폭발적인 증가와 네트워크 관련 기술의 발전으로 다양한 분야에서 네트워크를 기반으로 하는 응용 서비스가 등장하고 있다. 이러한 응용 서비스는 시간과 장소에 제한되지 않으면서도 일시적으로 응용 서버의 실패나 특정 호스트와의 네트워크 단절이 발생되는 상황에서도 클라이언트 측에 투명하고 지속적으로 동일한 서비스를 제공할 필요가 있다.

JACE 시스템은 기본적으로 네트워크 단절과 응용 서버의 실패를 고려한 시스템으로서 자바 응용 서비스가 프로세스 그룹의 형태로 동작될 수 있도록 지원한다. 그룹통신 시스템 모델은 크게 VS(Virtual Synchrony)[13, 14] 모델과 EVS(Extended Virtual Synchrony)[15] 모델로 구분된다. VS 모델은 그룹의 멤버들간에 송수신 되는 일반 메시지와 그룹 멤버십 변경 메시지를 순서화(ordering) 한다. VS 모델은 프로세스 P, Q가 멤버십 V_1에서 멤버십 V_2로 진행해 나갈 때 두 프로세스는 멤버십 V_1과 멤버십 V_2 사이에서 동일한 메시지들을 동일한 순서로 응용 계층에 전달할 수 있도록 보장한다. VS 모델을 확장한 EVS 모델은 L. E. Moser, Y. Amir, P. M Melliar-Smith 그리고 D. A. Agarwal에 의해 정형화되었다. EVS 모델은 하나의 프로세스 그룹이 일시적인 네트워크 단절로 상호 통신할 수 없는 두 개 이상의 서브그룹으로 분리되고 이후 다시 네트워크가 복원되어 상호 통신 가능할 때, 하나의 그룹으로 동작할 수 있도록 프로세스 멤버들간의 상태를 일관성 있게 유지한다. JACE 시스템은 EVS 모델을 지원한다.



(그림 3) JACE 시스템 구조

JACE 시스템 구조는 (그림 3)에서 보는 바와 같이 다수의 계층으로 구분된다.

- UDP 계층 : UDP 프로토콜을 사용하여 메시지를 송수신한다. 메시지는 단일캐스트 메시지와 다수의 수신자를 대상으로 하는 멀티캐스트 메시지로 구분될 수 있다. 이 계층에서 전달되는 메시지는 분실되거나 중복될 수 있다.
- 실패탐지(Failure Detect) 계층 : 네트워크의 단절과 프로세스 멤버의 실패를 발견하고 이를 상위계층으로 통지한다. 각 멤버마다 이웃한 멤버와의 소켓연결을 유지하며, 소켓연결에 예외가 발생할 때 이웃한 멤버의 실패를 의심하고, 이를 확인하기 위해 메시지를 전달한다. 응답이 없으면 실패로 간주하여 상위 계층으로 멤버의 실패를 통보한다.
- 단일캐스트/멀티캐스트 계층 : 그룹의 멤버에게 메시지를 송신하고 메시지의 수신자로부터 수신확인(ack) 메시지가 있었는지 파악한다. 일정시간(타임아웃) 동안 메시지를 수신하였다는 응답이 없으면, 메시지를 재 전송한다.

메시지를 재전송을 하였지만 여전히 수신확인 응답이 없으면 네트워크 단절이나 멤버 프로세스의 실패(crash)로 판단하고 상위 계층으로 멤버십 변경을 통보한다.

- 토큰기반 전체 순서화(Total Ordering) 계층 : 멤버들간에 발생되는 메시지들에 대하여 전체적으로 일관된 메시지 순서를 부여한다. 이를 위해 그룹에 참여하는 프로세스 멤버들을 대상으로 가상 링(virtual ring)을 형성하고, 토큰 메시지를 순환시킨다. 토큰을 받은 멤버만이 새로운 메시지에 시퀀스 번호를 지정하고 멤버들에게 전달할 수 있다. 이를 통하여 일련의 메시지 순서를 만든다.
- 멤버십 관리 계층 : 하위 계층으로부터 전달받은 실패발견 정보를 기초로 새로운 멤버십을 형성하며, 새로운 멤버가 그룹에 참여하거나 탈퇴가 발생할 때 이를 하위 계층으로 통지하고 멤버십을 변경한다.
- 상태전이 계층 : 새로운 멤버가 그룹에 참여하거나 네트워크 단절로 상호통신 할 수 없었던 멤버가 네트워크 복원으로 다시 동일 그룹으로 참여할 때 프로세스 멤버들간의 상태정보를 동일하게 맞추기 위해 상태복원 알고리즘을 수행한다.

JACE 시스템은 그룹을 형성하는 응용 서버의 요구사항에 따라 위의 계층을 조합할 수 있다. 예를 들어, 낮은 신뢰성이 요구된다면 UDP, 실패탐지, 단일캐스트, 멤버십 관리 계층만을 조합으로 그룹통신의 성격을 지정할 수 있으며, 높은 신뢰성이 요구된다면 UDP, 실패탐지, 단일캐스트, 토큰기반 Total 계층, Safe 계층, 멤버십 관리 계층, 상태전이 계층 등의 조합으로 그룹통신의 신뢰성을 높일 수 있다. 따라서 JACE 시스템을 기반으로 응용 프로그램을 개발함으로써 응용 서버의 가용성을 높일 수 있다.

2.3 그룹통신을 이용한 응용 사례

〈표 2〉 그룹통신 시스템의 응용 사례

그룹통신 시스템	응 용 사 례
Transis	• SNMP 기반 네트워크 관리 툴킷 • 공유(shared) 화이트 보드 • 협력(collaboratoin) 작업 시스템의 메시징 기능과 멤버십 기능 • 멀티미디어 멀티캐스트 서비스
Horus	• CORBA request broker • 결함포용(fault-tolerant) 멀티미디어 툴킷 • 결함포용 웹 서버 • 공동 문서 편집기
Totem	• 항공 트래픽 제어 시스템(데모 버전) • Eternal System
Jgroup	• RMI 확장 • JINI 확장
JACE	• 중복 객체공간 서비스(Replicated Object Space Service) • LDAP 서버와 LDAP 서비스 제공자

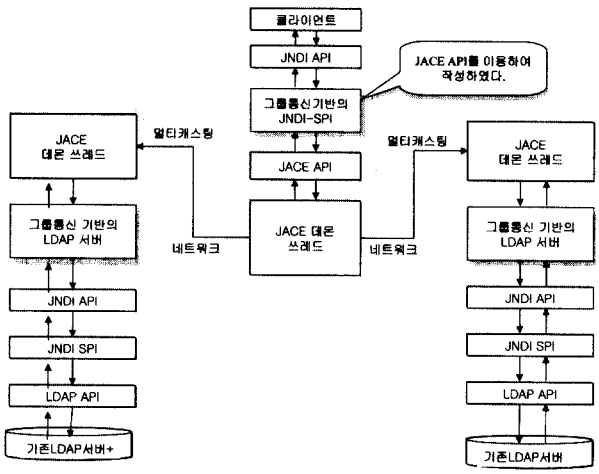
국내외에서 개발된 그룹통신 시스템의 응용 사례를 보면 <표 2>와 같다. 조사한 바에 의하면 Horus 시스템만이 실질적으로 여러 분야에 활용된 것을 알 수 있었다.

3. 그룹통신 기반의 LDAP 서버 설계

다수의 LDAP 서버들 사이에 디렉토리 정보를 중복하여 유지함으로써 특정 LDAP 서버와의 네트워크 단절이나 응용 서버의 실패(crash)와 같은 결함이 발생되는 상황에서도 투명하고 지속적으로 서비스를 제공하는 것이 바람직하다. LDAP 서버들 사이에 디렉토리 관련 정보를 동일하게 유지하고 프로세스 그룹으로 동작할 수 있도록 JACE 시스템의 그룹통신 기능을 이용한다.

3.1 그룹통신 기반의 LDAP 서버 모델

기존의 LDAP 클라이언트/서버 모델은 앞의 (그림 2)에서와 같이 클라이언트의 서비스 요청이 LDAP 서비스 제공자(JNDI SPI)를 통하여 직접적으로 LDAP 서버에 전달되고 응답을 받는 구조이다. 이러한 구조에서는 응용 서버의 실패나 클라이언트와 LDAP 서버 사이에 네트워크 단절이 발생할때 서비스를 제공받지 못하는 단점이 있다. 따라서 클라이언트는 이와 같은 상황에서도 투명하게 지속적으로 LDAP 디렉토리 서비스를 제공받는 것이 바람직하다. 본 논문에서 제안한 그룹통신 기반의 LDAP 클라이언트/서버 모델은 (그림 4)에서 보는 바와 같이 클라이언트 측의 서비스 제공자(*그룹통신 기반의 JNDI SPI*)가 그룹통신 시스템에 연결되어 클라이언트의 디렉토리 서비스 요청을 그룹통신 기반의 LDAP 서버에게 전달하는 구조이다.



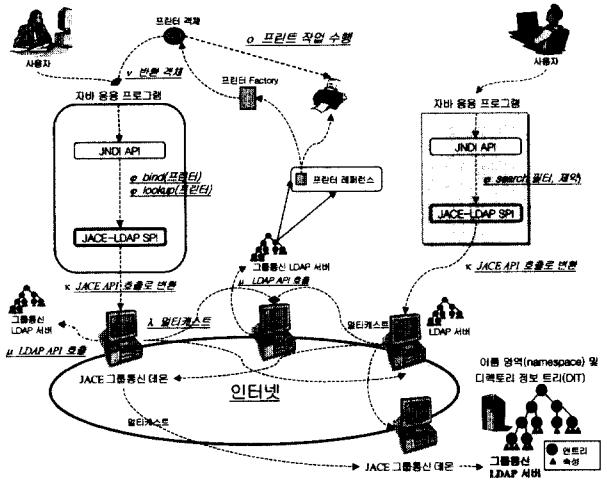
(그림 4) 제안된 그룹통신 기반의 LDAP 서버 모델

(그림 4)에서 이루어지는 수행 알고리즘은 다음과 같다.

- 1 단계 : 자바 응용 프로그램의 JNDI API 호출은 내부적으로 그룹통신 기반의 LDAP 서비스 제공자에게 전달된다.
- 2 단계 : 그룹통신 기반의 LDAP 서비스 제공자는 클라

이언트의 디렉토리 서비스 요청정보를 포함하는 그룹통신 이벤트 메시지로 변경한다.

- 3 단계 : 2 단계에서 생성된 메시지는 로컬 시스템 상의 그룹통신 데몬 쓰레드에 의해 상위 계층으로 부터 하위 계층으로 전달되며, 이때 각 계층에 관련된 헤더정보를 추가하여 네트워크 상으로 멀티캐스트 혹은 단일캐스트 된다.
- 4 단계 : 멀티캐스트 되어진 메시지는 다른 호스트 상의 그룹통신 데몬 쓰레드에 의해 수신되어지고, 상위 계층으로 전달된다. 그룹통신 시스템을 사용함으로써 메시지 전달에 대한 신뢰성을 보장받는다.
- 5 단계 : 최상위 계층으로 전달된 이벤트 메시지는 그룹통신 기반의 LDAP 서버에게 전달된다. 그룹통신 기반의 LDAP 서버는 이벤트 메시지로부터 클라이언트가 보낸 원래의 JNDI 호출 정보를 구한다.
- 6 단계 : 그룹통신 기반의 LDAP 서버는 클라이언트의 JNDI 호출을 대신하여 호출한다. 동일 호스트 상에 존재하는 JNDI 드라이버(LDAP SPI)를 이용하여 기존 LDAP 서버에게 디렉토리 서비스 요청을 전달한다.
- 7 단계 : 엔트리의 추가, 삭제, 수정에 대한 요청은 반환값을 필요로 하지 않지만, 엔트리 조회의 경우는 반환값이 있으므로 기존 LDAP 서버로부터 반환값을 구한다.
- 8 단계 : 그룹통신 기반의 LDAP 서버는 응답 받은 결과정보를 다시 클라이언트에게 전달하기 위해 2 단계에서 5 단계의 과정을 수행한다. (2 단계에서는 응답정보를 포함하는 이벤트 메시지로 변경한다.)
- 9 단계 : 응답정보를 포함하는 이벤트 메시지를 수신한 그룹통신 데몬 쓰레드는 클라이언트 측의 그룹통신 기반의 LDAP 서비스 제공자에게 그 결과를 반환한다.
- 10 단계 : 그룹통신 기반의 LDAP 서비스 제공자는 수신한 메시지로부터 응답결과를 구하여 최종적으로 클라이언트에게 반환한다.



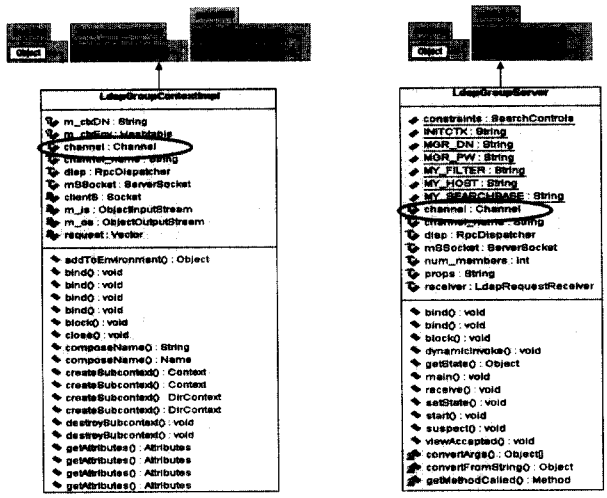
(그림 5) 그룹통신 기반의 LDAP 서버 사용 시나리오

사용자는 (그림 5)에서 보는 바와 같이 자바 응용 프로그램을 사용하여 네트워크 상에 존재하는 다양한 정보를 얻을 수 있다.

클라이언트 응용 프로그램 개발자는 기존의 JNDI 프로그래밍 방법과 동일하게 JACE 기반의 명명 서비스와 디렉토리 서비스를 이용하는 프로그램을 작성할 수 있다.

3.2 그룹통신 기반의 LDAP 서버와 서비스 제공자 클래스

(그림 6)은 그룹통신 기반의 LDAP 서버와 서비스 제공자(LDAP SPI)에 대한 주요 클래스인 LdapGroupServer 클래스와 LdapGroupContextImpl 클래스의 구조를 보여준다. 주요 특징으로는 이 두 클래스가 모두 그룹통신 데몬 쓰레드와 연결되어 동작하므로 멤버필드에 그룹통신과 연결을 유지하는 채널(channel) 필드를 가지고 있다. 또한 두 클래스 모두 MembershipListener와 MessageListener 인터페이스를 구현하였다. 이를 통해 그룹통신 데몬 쓰레드로부터 메시지를 수신하고, 멤버십의 변경을 통보 받는다.

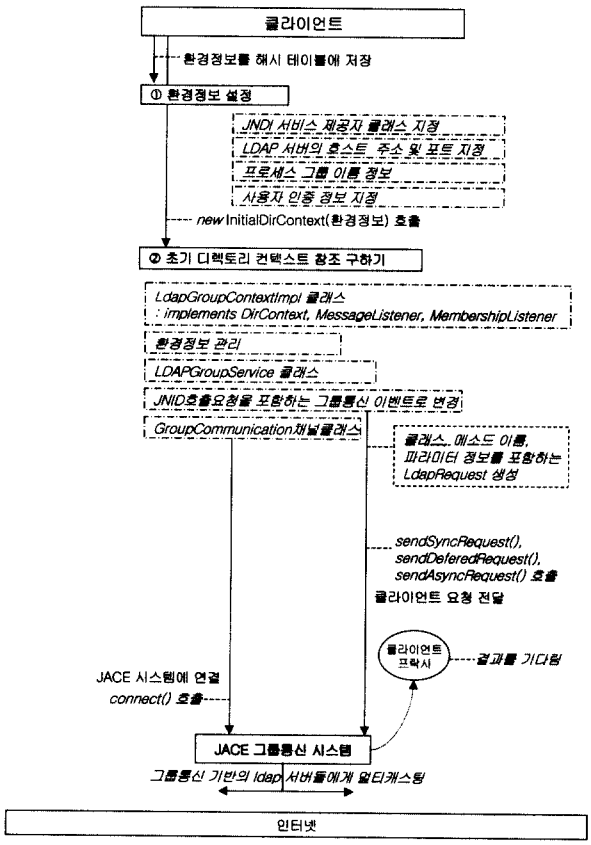


(그림 6) 그룹통신기반의 LDAP 서버와 LDAP SPI 클래스 구조

3.3 LDAP 서버에 연결하기

클라이언트는 우선 (그림 7-①)과 같이 연결하고자 하는 LDAP 서버의 주소, 포트 번호, LDAP 서비스 제공자 클래스, 사용자 인증 정보, 프로세스 그룹정보 등의 환경 정보를 해시 테이블에 저장한다. 일반 JNDI 응용 프로그램 작성과 동일하게 환경 정보를 기초로 하여 초기 디렉토리 컨텍스트에 대한 참조를 구하면 사실상 내부적으로 연결이 완성된다. LDAP 클라이언트-서버 모델과의 차이점은 그룹통신 기반의 LDAP 모델의 경우 클라이언트가 직접적으로 LDAP 서버와 연결되는 것은 아니다. 클라이언트는 프로세스 그룹의 형태로 동작하는 LDAP 서버들과 통신하기 위해서 중간에 JACE 그룹통신 시스템을 경유한다. LdapGroupContextImpl 클래스는 DirContext 인터페이스를 구현하였

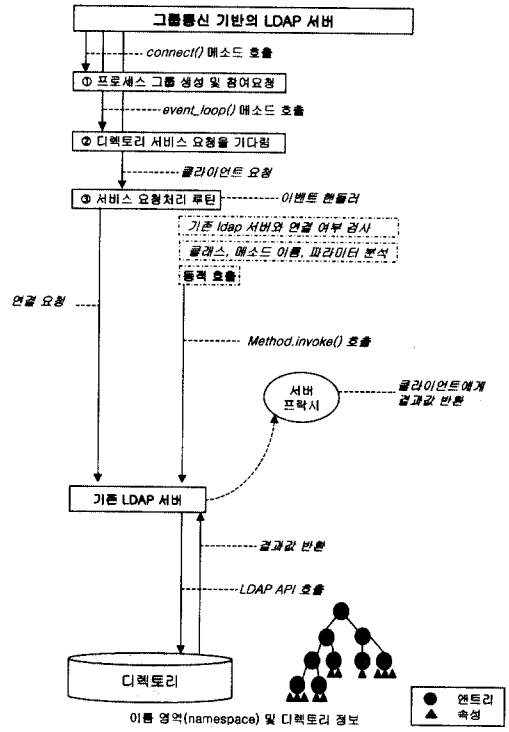
고, 환경정보를 포함한다. 내부적으로는 각 JNDI 함수 호출을 캡슐화한 LdapRequest 클래스, LDAP 서비스 요청을 그룹통신을 이용하여 멀티캐스팅 하기 위하여 그룹통신 메시지로 변환하는 클래스, JACE 시스템과의 실질적인 연결을 담당하는 GroupCommunicaton 클래스 등이 이용된다. 초기 디렉토리 컨텍스트의 참조를 구하는 요청(그림 7-②)은 JACE 시스템을 경유하여 그룹통신 기반의 LDAP 서버에게 이벤트 형태로 전달된다. 이벤트 핸들러에 의해 요청은 분석되어 LDAP 서버그룹에 클라이언트로서 참여하게 된다. 클라이언트는 이 과정을 마치고 나서 그룹통신 기반의 LDAP 서버를 이용할 수 있게 된다.



(그림 7) LDAP 서버에 연결 과정

3.4 LDAP 서버에 서비스 요청

그룹통신 기반의 LDAP 서버는 우선 (그림 8-①), (그림 8-②)와 같이 프로세스 그룹을 형성하기 위해 *connect*( ) 메소드를 호출한 후 클라이언트로부터 요청을 받기 위해 *event_loop*( )을 호출하여 대기 상태에 놓이게 된다. 클라이언트로부터 발생되는 LDAP 디렉토리 서비스에 대한 각 요청은 (그림 8-③)과 같이 그룹통신 기반의 LDAP 서버의 이벤트 핸들러에 의해 이벤트에 포함된 요청 정보가 분석되며, 자바의 동적 호출 기능을 이용하여 LDAP 서버에게 전달된다.



(그림 8) LDAP 서비스 요청 과정

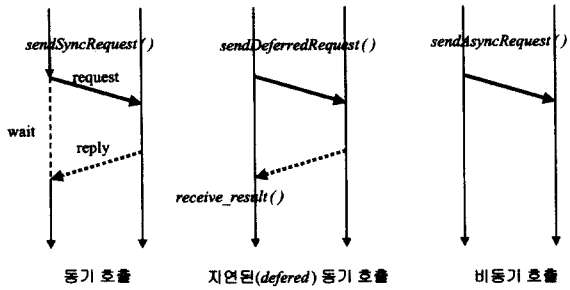
4. 그룹통신 기반의 LDAP 서버 구현

이 장에서는 클라이언트의 디렉토리 서비스 요청이 반환값을 필요로 하는가에 따라 구분되는 동기 호출과 비동기 호출에 대하여 살펴보고, 그룹통신 기반의 견고한 LDAP 서버의 주요 코드에 대하여 설명한다.

4.1 비동기 호출과 동기 호출

LDAP 디렉토리 서비스가 프로세스 그룹의 형태로 서비스를 제공하기 위해서는 클라이언트가 그룹내의 LDAP 서버들에게 서비스 요청을 보내고 그 결과를 되돌려 받을 수 있는 방법이 제공되어야 한다. 클라이언트의 요청은 해당 프로세스 그룹에게 요청에 관련된 정보를 포함하는 이벤트를 전달하는 형태로 이루어지며 일반적으로 비동기(asynchronous) 호출과 동기(synchronous) 호출로 구분된다. 동기 호출은 결과가 반환될 때까지 호출은 블록(block)되며 비동기 호출은 결과를 기다리지 않고 곧바로 계속하여 다른 작업을 수행한다. 동기호출은 결과값을 수신할 때까지 블록되는 호출(*sendSyncReqeust*)과 명시적으로 결과 값 수신을 위해 별도의 메소드(*received_result*( ))를 호출해야하는 지연된 동기호출(*sendDeferedRequest*)로 구분된다.

JACE 시스템에서는 서비스 요청에 관련된 정보를 포함하는 메시지를 그룹의 멤버들에게 전달하기 위해 세 가지 유형의 이벤트 전달방법을 정의하였다. 아래 (그림 9)은 이들의 차이점을 보여주고 있다.



(그림 9) 동기화 유무에 따른 이벤트 발생 유형

4.2 그룹통신을 이용한 LDAP 서버의 구현

응용 프로그램 개발자는 JACE 시스템이 제공하는 API를 이용하여 프로세스 그룹의 형태로 동작하는 자바 응용 서비스를 개발할 수 있을 뿐만 아니라 자바로 작성된 기존의 응용 프로그램도 JACE 시스템을 기반으로 하는 응용 서비스로 쉽게 전환이 가능하다. 4.2절에서는 앞에서 제시한 API를 이용하여 프로세스 그룹으로 동작하는 LDAP 응용 서버의 구현 대하여 살펴본다. 본 시스템은 JDK1.4와 넷스케이프 디렉토리 서버 4.1을 이용하여 개발되었다.

아래의 코드는 그룹통신 기반의 LDAP 서버 클래스에 대한 프로그램 코드이다.

```
// LdapGroupServer.java
package ldapgroup ;

import java.lang.reflect. * ;   // 동적 메소드 호출을 위해 reflection
                                을 사용한다.
import pslab.GoD. * ;
import pslab.GoD.blocks. * ;
// 코드 생략 : 헤더파일 포함

public class LdapGroupServer implements MembershipListener,
MessageListener{

    // 기존 LDAP 드라이버 사용을 위한 필드
    public static String INITCTX =
          "com.sun.jndi.ldap.LdapCtxFactory" ;

    public static String MY_HOST = "ldap : //Mylife : 389" ;
    //  기존 LDAP 서버 url
    // 기타 멤버 필드 …

    LdapRequestReceiver mReceiver ;   // 클라이언트 요청을 담
                                      당하는 쓰레드
    ServerSocket mSSocket ;

    Channel mGroupCommunication ;     // 그룹통신 데몬 쓰레
                                      드와의 연결 담당
    ReponseDispatcher mRspDispatcher ; // 응답 메시지 처리자
    final String mGroupName = "LdapGroups" ;
    // 참여 그룹 이름
    int mNumOfMembers = 1 ;           // 그룹의 최소 멤버 수
    // 그룹통신 데몬 쓰레드의 사용프로토콜 지정
    String mGroupProps = "UDP : PING : FD : UNICAST :
                          FRAG : TOTAL_TOKEN : " +
          "MEMBERSHIP : STATE_TRANSFER : " ;

    public static void main (String args[ ]) {
    LdapGroupServer ldap_server = new
     LdapGroupServer( ) ;
        try{
         ldap_server.start( ) ;
        } catch(Exception e){ }
    }
    // 동적 메소드 호출을 수행한다.
    public static void dynamicInvoke (String args[ ]) throws
    Exception
    {
        Class myClass = Class.forName (args[0]) ;
        Method meth = getMethodCalled (myClass, args[1],
                      args.length-2) ;
        Object target ;

        if ((meth.getModifiers( ) & Modifier.STATIC) != 0) {
            target = null ;
        } else {
            target = myClass.newInstance() ;
        }
        String newArgs[ ] = new String[args.length - 2] ;
        for(int i = 0 ; i < newArgs.length ; i++) {
            newArgs[i] = args[i+2] ;
        }
        Object result = meth.invoke(target, convertArgs
                        (newArgs, meth.getParameterTypes())) ;

        if (result != null) {
            System.out.println(result.toString( )) ;
        }
        if (target != null) {
            System.out.println(target.toString( )) ;
        }
    }

    private static Object[ ] convertArgs(String args[ ], Class
    types[ ]) throws Exception {
        Object[ ] returnArgs = new Object[args.length] ;
        for(int i = 0 ; i < args.length ; i++) {
            returnArgs[i] = convertFromString(args[i], types[i]) ;
        }
        return returnArgs ;
    }
    private static Object convertFromString (String value, Class
                                        type) throws Exception{
        if (type == Character.TYPE) {
            return new Character (value.charAt(0)) ;
        } else if (type == Integer.TYPE) {
            return new Integer (value) ;
        } else if (type == Boolean.TYPE) {
            return new Boolean (value) ;
        } else if (type == String.class) {
            return value ;
        }
        // otherwise, try and create it from a String
        Class [ ] params = { String.class } ;
        Object [ ] args = { value } ;
        return type.getConstructor (params).newInstance(args) ;
    }
    private static Method getMethodCalled (Class myClass, String
    name, int length) {
        /* 코드 생략 */
    }
```

```
    public void viewAccepted(View new_view) { … }

    public void start( ) throws IOException{
    try {
        mGroupCommunication = new GroupCommunication
        (mGroupProps) ;
        mGroupCommunication.setOpt (Channel.
        GET_STATE_EVENTS, new Boolean(true)) ;
     mGroupCommunication.SetOpt (Channel.LOCAL,
        new Boolean(false)) ;
        mRspDispatcher = new ResponseDispatcher
        (mGroupCommunication, this, this, this) ;
        mGroupCommunication.connect(mGroupName) ;
        // 그룹에 연결하기
    }
    catch(Exception e) {
        System.err.println ("LdapGroupServer.Start() : " + e) ;
        System.exit(-1) ;
    }
    mGroupCommunication.getState (null, 0) ;

    // event_loop( ) ;
    }

    public void suspect (Address suspected_mbr) { … }
    public void block() { … }
    public void receive(Message msg) { … }
    public Object getState() { … }
    public void setState(Object state) { … }

    public void bind(Vector request){

    try {
            Hashtable env = new Hashtable() ;
            env.put(Context.INITIAL_CONTEXT_FACTORY,
                    INITCTX) ;
            env.put(Context.PROVIDER_URL, MY_HOST) ;
            env.put(Context.SECURITY_AUTHENTICATION,
                    "simple") ;
            env.put(Context.SECURITY_PRINCIPAL, (String)
                    request.elementAt(1)) ;
            env.put(Context.SECURITY_CREDENTIALS,
                    (String) request.elementAt(2)) ;

            DirContext ctx = new InitialDirContext(env) ;

            ctx.bind ((String) request.elementAt(3),
                    request.elementAt(4)) ;

        }catch(Exception ex){
        }
    }

    public void bind(String request){ /* 코드 생략 */ }

}
```

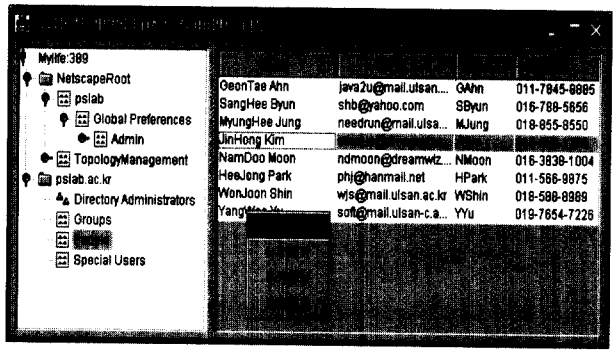
그룹통신 기반의 LDAP 서버는 멤버십 변경 메시지와 일반 메시지를 수신하기 위하여 MembershipListener, MessageListener 인터페이스를 구현한다.

그룹통신 기반의 LDAP 자바 응용 서버는 시작시 JACE와의 연결을 이룬 후 자신이 원하는 프로세스 그룹을 생성하거나 기존의 그룹에 참여할 수 있다. JACE 시스템과 연결하고 LDAP 서버는 클라이언트의 디렉토리 서비스 요청을 기다리기 위해 event_loop() 메소드를 호출한다.

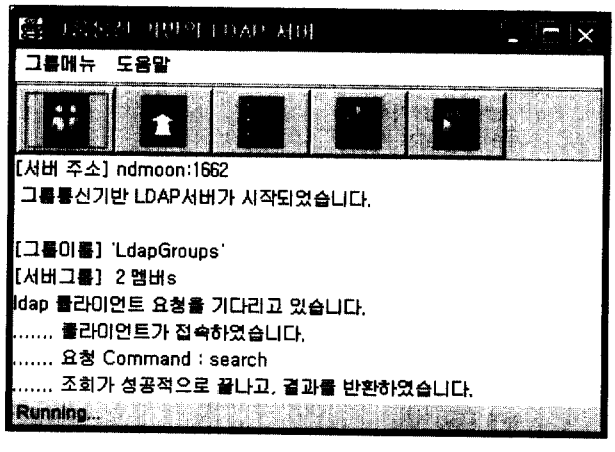
그리고 위의 LDAP 자바 응용 서비스는 클라이언트의 서비스 요청을 포함하는 이벤트를 수신할 수 있도록 receive() 메소드를 정의하였다. MessageListener 인터페이스에 선언된 receive() 메소드는 그룹 데몬 쓰레드로부터 전달되는 이벤트를 수신하고 처리하기 위한 이벤트 핸들러이다. 이 메소드를 구현함으로써 LDAP 서버는 그룹으로 전달되는 모든 이벤트를 처리할 수 있다. 이 핸들러는 JACE 시스템에 의해 독립된 쓰레드 형태로 수행된다. 따라서 수신 메시지 핸들러를 정의한 응용 서버가 sendDeferedRequest() 나 sendSyncRequest()에 의해 블록되더라도 응용 서비스의 수행상태와는 관계없이 해당 이벤트를 처리할 수 있게 된다.

4.3 실행 결과 및 평가

(그림 10)은 그룹통신 기반의 LDAP 서버로부터 디렉토리 서비스를 이용하는 간단한 클라이언트 예제 프로그램이다. 그림에서 왼쪽부분은 디렉토리의 컨텐츠를 트리형태로 보여주고 있으며, 오른쪽 화면은 왼쪽 트리에서 선택된 노드가 포함하고 있는 엔트리에 대한 각각의 속성을 보여주고 있다. 엔트리 추가, 삭제, 수정, 조회 기능은 팝업 메뉴 형태로 제공하고 있다.



(그림 10) 그룹통신 기반의 LDAP 클라이어트 실행화면



(그림 11) 그룹통신 기반의 LDAP 서버 실행 화면

(그림 11)은 그룹으로 동작하는 그룹통신 기반의 LDAP

〈표 3〉 성능 평가

명령어	1개의 요청		10개의 요청		100개의 요청		1000개의 요청	
	기존 ldap서버	그룹통신 ldap서버	기존 ldap서버	그룹통신 ldap서버	기존 ldap서버	그룹통신 ldap서버	기존 ldap서버	그룹통신 ldap서버
엔트리 조회	16ms	200ms	110ms	1093ms	578ms	7141ms	4640ms	64890ms
엔트리 추가 엔트리 삭제 엔트리 수정	156ms	422ms	1203ms	4203ms	7406ms	54000ms	107266ms	571500ms

서버의 실행화면이다. 그림에서 보는 바와 같이 간단한 메뉴와 도구모음으로 구성되어 있으며, 클라이언트의 요청 정보와 현 서버의 상태를 화면에 출력하고 있다. 그림에 출력된 메시지를 살펴보면, 그룹으로 동작하고 있는 LDAP 서버가 2개이며, 클라이언트로부터 조회요청이 있었고, 응답을 전송하였음을 볼 수 있다.

<표 3>은 본 논문에서 개발한 그룹통신 기반의 LDAP 서버와 일반 LDAP 서버를 대상으로 성능평가를 측정한 결과를 보여준다. 클라이언트의 서비스 요청 횟수를 1회, 10회, 100회, 그리고 1000회로 구분하고 첫 요청이 발생한 시간에서 마지막 요청이 처리될 때까지의 경과한 시간 간격을 측정하였다. <표 3>에서 보는 바와 같이 디렉토리 서비스는 데이터의 변경보다는 데이터 조회에 더 적합한 시스템임을 알 수 있다. 엔트리 조회의 경우 그룹통신 기반의 LDAP 서버가 기존의 서버보다 10배 이상 느리고, 엔트리 변경의 경우 4배 이상 느린 결과를 보였다. 속도 저하의 문제점은 그룹통신 기반의 LDAP 서버와 기존의 실제 LDAP 서버를한 프로세스로 통합함으로써 상당정도 해결할 수 있을 것이다. 또한 LDAP 응용에 맞게 그룹통신 계층을 최적화 함으로써 시간비용을 줄 일 수 있을 것이다. 그룹통신을 이용한 LDAP 서버가 요청을 처리하는데 많은 시간이 소요되었지만, 특정 서버가 실패(crash)되더라도 다른 호스트상의 서버로부터 서비스를 제공받을 수 있었다. 윈도우 2000 서버를 설치한 펜티엄Ⅳ 1.7GHz 256MB와 펜티엄Ⅳ 1.2GHz 512MB에서 측정하였다.

<표 3>의 데이터 값은 클라이언트의 서비스 요청이 처리되기까지의 소요된 시간을 1000분의 1초로 표현한 것이다.

6. 결 론

LDAP 디렉토리 서비스는 인터넷이나 인트라넷 등 네트워크 상에 있는 파일이나 장치들과 같은 자원 등의 위치를 찾을 수 있도록 정보를 제공한다. 이러한 서비스를 제공하는 응용 서버는 네트워크의 단절과 같은 전산망의 오류가 발생되더라도 안정적이고 지속적인 서비스를 제공하는 것이 바람직하다. 이를 지원하려면 동일한 서비스를 제공하는 응용 서버들을 프로세스 그룹으로 동작시킬 수 있으며, 그룹 구성원들의 상태를 일관되게 유지해주는 그룹통신 시스템이 필요하다.

JACE는 자바 응용 서비스 그룹이 일시적인 네트워크 단절로 상호 통신할 수 없는 두 개 이상의 구성요소로 분리되더라도 지속적인 서비스를 제공하고, 네트워크가 복구되면 이전과 같이 하나의 그룹으로서 동작할 수 있도록 그룹 구성원의 일관성을 유지하는 EVS(Extended Virtual Synchrony) 모델을 지원하여 자바 응용 서비스의 신뢰성과 가용성(availability)을 높여주는 그룹통신 시스템이다.

본 논문에서는 JACE 시스템을 이용하여 동일한 서비스를 제공하는 다수의 LDAP 디렉토리 서비스가 프로세스 그룹으로 동작될 수 있는 그룹통신 기반의 LDAP 디렉토리 서버와 서비스 제공자의 설계 및 구현에 관하여 살펴보았다. 그룹통신 기반의 LDAP 서비스 제공자는 클라이언트의 서비스 요청을 파악하여 관련된 클래스 이름, 메소드 이름, 인자값, 정적 메소드 여부 등의 관련된 정보를 JACE 시스템을 이용하여 프로세스 그룹에 참여하고 있는 각 LDAP 서버에게 멀티캐스트/단일캐스트 한다. LDAP 프로세스 그룹에 참여한 각 서버들은 전달받은 요청을 분석하여 동적으로 LDAP 서버의 기능을 호출한다.

디렉토리 서비스를 제공하는 특정 호스트와의 네트워크의 단절과 같은 전산망의 결함이 발생되더라도 개발된 시스템을 통하여 클라이언트는 프로세스 그룹으로 동작하는 다른 LDAP 서버로부터 투명하게 디렉토리 정보를 제공받게 된다.

앞으로 가상의 네트워크 단절을 시뮬레이션하고, LDAP 서버의 상태를 모니터링을 할 수 있는 시뮬레이터를 개발하여 네트워크 단절 상황에 따른 시스템의 성능에 대하여 통계적인 자료를 산출할 계획이다. 현 시스템에서는 그룹통신 기반의 LDAP 서버와 기존의 LDAP 서버가 분리되어 존재하는 데, 이들의 기능을 하나의 프로세스로 통합할 계획이다.

참 고 문 헌

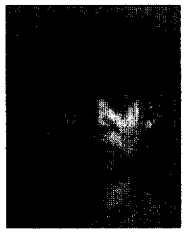
[1] W. Yeong, T. Howes and S. Kille, "Lightweight Directory Access Protocel," RFC 1777, March, 1995.

[2] Wahl, M., T. Howes & S. Kille, "Lightweight Directory Access Protocol (v3)," RFC2251, December, 1997.

[3] 문남두, 최혁재, 유양우, 박양수, 이명준, "자바를 이용한 Extended Virtual Synchrony의 지원", 정보과학회 추계학술발표논문집, 제25권 제2호, pp.409-411, 1998.

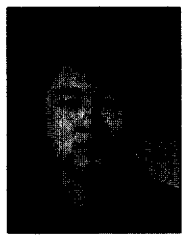
[4] 최혁재, 문남두, 박양수, 이명준, "자바 응용 서비스 개발을

위한 JACE 프로그래밍 인터페이스", 한국정보과학회 '99 봄 학술발표논문집, 제26권 제1호, pp.382-384, 1999.
[5] 문남두, 안건태, 유양우, 이명준, "JACE : 인터넷 환경을 지원하는 신뢰성 있는 그룹통신 시스템", 정보처리논문지, 제6권 제11호, pp.3379-3389. 1999.
[6] Sun Microsystems, Inc. Java Naming and Directory InterfaceTM. Application Programming Interface (JNDI API), July, 1999.
[7] Sun Microsystems, Inc. Java Naming and Directory InterfaceTM Service Provider Interface(JNDI SPI) Standard Edition, v1.3, July, 1999.
[8] Sun Microsystems, Inc. JNDI Implementor Guidelines for LDAP Service Providers Draft 0.2.
[9] D. Malki, Multicast Commnication for High Availability. Ph.D. thesis, Institute of Computer Science, The Hebrew University of Jerusalem, Israel, 1994.
[10] L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, R. K. Budhia, and C. A. Lingley-Papadopoulos, "Totem : A Fault-Tolerant Multicast Group Communication System," Communications of the ACM, Vol.39 No.4, pp.54-63, 1996.
[11] B. Ban, JavaGroups User's Guide, Technical report, Cornell University, Aug., 1999.
[12] A. Montresor, System Support for Programming Object-Oriented Dependable Applications in Partitionable Systems, PhD thesis, Department of Computer Science, University of Bologna, July, 2000.
[13] K. P. Birman, "Virtual Synchrony Model," In Reliable Distributed Computing with the Isis Toolkit, IEEE press.
[14] K. Birman and T. Joseph, "Exploiting Virtual Synchrony in Distributed Systems," In *Proceeding of the ACM Symposium on Operating Systems Principles*, pp.123-138, November, 1987.
[15] L. E. Moser, Y. Amir, P. M. Melliar-Smith and D. A. Agarwal, "Extended Virtual Synchrony," In Proceeding of the 14th Internatinal Conference on Distributed Computing Systems, pp.56-65, June, 1994.
[16] Mark Wilcox, "Implementing LDAP," Wrox Press Ltd., pp.269-298, 1999.
[17] Rob Weltman, Tony Dahbura, "LDAP Programming with Java," Addison-Wesley, 2000.



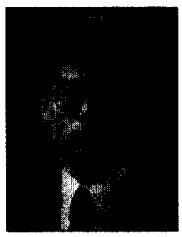
문 남 두

e-mail : ndmoon@dreamwiz.com
1997년 울산대학교 전자계산학과(공학사)
1999년 울산대학교 컴퓨터정보통신 공학부 (공학석사)
1999년~현재 울산대학교 컴퓨터정보통신 공학부 박사과정
관심분야 : 그룹통신 시스템, 분산객체 시스템, CSCW 등



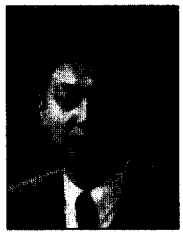
안 건 태

e-mail : java2u@lycos.co.kr
1999년 울산대학교 전자계산학과(학사)
2001년 울산대학교 컴퓨터정보통신 공학부 (석사)
2001년~현재 울산대학교 컴퓨터정보통신 공학부 박사과정
관심분야 : 그룹웨어, 이동에이전트시스템, 생물정보학 등



박 양 수

e-mail : yspk@uou.ulsan.ac.kr
1978년 울산대학교 전자계산학과(학사)
1981년 서울대학교 계산통계학과(석사)
1986년~현재 서울대학교 계산통계학과 박사과정
1980년~현재 울산대학교 컴퓨터정보통신 공학부 근무(부교수)
관심분야 : 분산처리, 컴퓨터알고리즘 등



이 명 준

e-mail : mjlee@mail.ulsan.ac.kr
1980년 서울대학교 수학과(학사)
1982년 한국과학기술원 전산학과(석사)
1991년 한국과학기술원 전산학과(박사)
1993년~1994년 미국 버지니아대학 교환교수
1982년~현재 울산대학교 컴퓨터정보통신 공학부 교수
관심분야 : 프로그래밍언어, 분산 객체 프로그래밍 시스템, 병행 실시간 컴퓨팅, 인터넷 프로그래밍시스템 등