Non-Java 장치를 지원하기 위한 Jini 서로게이트 시스템의 설계 및 구현

(Design and Implementation of Jini Surrogate System for Supporting Non-Java Devices)

최 현 석 † 모 상 덕 †† 정 광 수 ††† 오 승 준 †††

(Hyunseok Choi) (Sangdok Mo) (Kwangsue Chung) (Seoungjun Oh)

요 약 최근 내장형 장치를 인터넷과 연결하고자 하는 요구가 늘어나고 있다. 이러한 정보기기들 간의 동적인 분산 네트워크를 구성하는 기술로서 Jini 기술이 주목을 받고 있다. 그러나, 소형 장치에 Jini 기술을 적용하여 서비스를 제공하기 위해서는 높은 하드웨어 사양을 요구하는 문제점이 지적되고 있다. 본 논문에서는 이러한 non-Java 장치를 Jini 네트워크에 접속하여 Jini 서비스를 제공하기 위한 Jini 서로게이트 시스템(Surrogate System)과 non-Java 장치간의 서로게이트 프로토콜(Surrogate Protocol)을 설계하고 자바 언어로 구현하였다. 서로게이트 시스템은 장치가 Jini 네트워크에 연결되어 Jini 서비스를 제공하기 위해 필요한 Discovery와 Join 과정을 대리해주는 역할을 수행한다. Jini 클라이언트는 서로게이트 시스템을 통해 장치가 제공하는 서비스를 이용할 수 있다. 구현한 서로게이트 시스템이 정상적으로 동작하는지 검증하기 위해 Jini 예제 프로그램을 만들어 시험하였다. 시험결과, 구현한 서로게이트 시스템을 이용해 Jini 클라이언트가 non-Java 장치의 서비스를 이용할 수 있음을 확인하였다.

키워드 : 지니, 자바, 서로게이트, 내장형 시스템

Abstract Recently, there has been increasingly demand for connecting a embedded device to the Internet. Jini technology is interested in automatically composing a distributed network with devices. But, there are some problems that the device needs high hardware requirements to adopt Jini technology for supporting Jini-enabled services. In this paper, we focused on design and implementation of surrogate system that supports non-Java devices in Jini networks. This system and protocol are implemented in Java language. The surrogate system delegates Discovery and Join processing to support a Jini service in connected networks. A Jini client can use service of the device through the surrogate system. We tested a Jini sample program to verify the implemented surrogate system. In the test result, we showed that the Jini client can use functionalities and operations of the non-Java device through the surrogate system.

Key words : Jini, Java, Surrogate, Embedded System

1. 서 론

최근 인터넷이 발전하면서 정보기기 및 장치들을 네트워크로 연결하여 인터넷이 연결된 어디에서나 서비스를 이용하려는 요구가 늘어나고 있다. 이러한 장치와 소프트웨어를 분산 네트워크로 통합하기 위해 여러 가지 분산 객체 기술들이 개발되었다.

현재 분산 환경을 제공하는 대표적인 미들웨어로는 OMG(Object Management Group)의 CORBA (Common Object Request Broker Architecture)와 Microsoft의 DCOM(Distributed Component Object Model) 등의 기술이 있으나, 정보기기 및 장치에 적용하기에는 많은 문제점을 가지고 있다[1,2]. 이러한 정보

† 비 회 원 : 삼성전자 연구원
harrisonchoi@samsung.com
†† 비 회 원 : 광운대학교 전자통신공학과
sdmo@explore.kwangwoon.ac.kr
††† 종신회원 : 광운대학교 전자공학부 교수
kchung@daisy.kwangwoon.ac.kr
sjoh@media.gwu.ac.kr
논문접수 : 2000년 12월 14일
심사완료 : 2002년 7월 26일

기기나 장치의 분산 환경을 위해 Sun Micro systems에서 사무실 및 가정내의 정보기기를 통합 네트워크로 연결하기 위한 자바 기반의 분산 네트워크 미들웨어인 Jini 기술이 개발되었다[3,4]. Jini 기술은 네트워크의 하부구조로 네트워크상의 소프트웨어와 하드웨어를 하나의 서비스로 통합해주며, 별도의 인위적인 조작이나 설치과정 없이 네트워크의 플러그 앤 플레이(Network Plug and Play)를 가능하게 해주는 프로토콜이며 클래스 라이브러리이다.

그러나, 디지털 온도계와 같은 대부분의 소형 장치들은 낮은 프로세서 수행속도와 적은 양의 메모리 환경 및 다양한 네트워크 연결 환경을 갖고 있다. Jini 기술이 가능하려면 시스템에 많은 오버헤드를 부가하는 RMI (Remote Method Invocation)와 자바 가상 머신(Java Virtual Machine)환경을 갖추어야 하기 때문에 대부분의 소형 장치에 Jini 기술을 채택하기가 어렵다[6]. Jini 기술이 최근 홈 네트워크(Home Network)의 미들웨어로 주목받고 있는데 이러한 정보가전(Information Appliance) 및 소형 장치들을 Jini 네트워크로 통합하여 장치의 서비스를 이용하고자 하는 클라이언트들에게 서비스를 제공하고자 한다. 그러나, 아직 대부분의 장치들이 많은 오버헤드를 갖는 Jini 기술을 탑재하지 못하고 있기 때문에 Jini 네트워크에 연결되어 서비스를 제공하지 못하고 있으며, 시장에서도 기술을 적용한 장치 제품을 찾아보기 어려운 실정이다[7].

내장형 장치에서 Jini가 갖는 문제점을 해결하기 위해 Jini 네트워크에서 Jini의 Discovery 및 Join 역할을 대리해주는 서로게이트 방식이 가장 현실적인 방법이라고 할 수 있겠다. 서로게이트 호스트를 통해서 Jini 클라이언트에게 non-Java 장치의 서비스를 투명하게 제공하는 시스템이 서로게이트 시스템(Surrogate System)이다[10,11].

본 논문에서는 non-Java 장치의 Jini 서비스를 Jini 네트워크에 등록하고 non-Java 장치와 Jini 클라이언트 간의 중계역할을 담당하는 서로게이트 시스템을 설계하고 구현하였다. 본 서로게이트 시스템은 서로게이트 프로토콜을 사용하여 서로게이트 호스트와 장치간의 존재를 파악하는 Discovery Manger 모듈, IP 망을 위한 IP Interconnect Adapter 모듈 및 서비스 프락시 코드(Service Proxy Code)를 다운받는 Surrogate Retriever 모듈로 구성된다.

2. 관련 연구

2.1 Jini 기술

Jini는 Sun Microsystems에서 제안하고 있는 차세대 접속기술로서, 자바를 이용하여 네트워크상의 기기나 소프트웨어를 동적으로 연계시키는 분산 객체 기술이다. 특정한 컴퓨터의 하드웨어와 소프트웨어에 상관없이 Jini를 채택하고 있는 기기들이 네트워크에 연결되기만 하면, 외부 설정이나 별도의 설치 절차 없이 서비스를 사용할 수 있다. 또한 Jini 기술을 통해 네트워크 상의 서비스 및 클라이언트간의 상호작용을 통해 여러 가지의 작업을 할 수 있도록 하는 기술이다.

Jini를 장착한 기기들은 하나의 서비스(Service)의 개념으로 처리되는데 이러한 서비스들이 네트워크를 통해 상호 연결되어서 서로 서비스를 공유할 수 있는 Jini 연합체(Federation)를 형성하게 된다. 여기서 Jini 연합체는 Jini를 장착한 기기끼리 특정한 업무를 실행 및 처리하기 위해 필요한 서비스의 집합으로서, 서로간에 상호 서비스를 이용할 수 있도록 해주는 기능을 가지고 있다.

Jini 기술은 네트워크에 접속할 수 있는 모든 기기에 대해서 적용할 수 있다. 이와 같이 Jini 기술을 이용하여 Jini 시스템을 구축하고자 할 때는 다음과 같은 조건을 갖추어야 한다. Jini 기술을 장착한 기기에 어떤 형식으로든 전원이 공급되어져야 하고, 랜·무선·전화선·전용선 등의 형식으로 네트워크에 접속될 수 있어야 한다. 마지막으로 운영체제와 더불어 Jini는 자바 프로그래밍 언어의 특성을 이용하여 구현되었기 때문에 자바가상머신이 갖추어져야 한다. 그림 1은 Jini 서비스 이용자가 Jini의 Lookup 서비스를 통해 Jini 서비스를 이용하는 과정을 나타낸 Jini 네트워크이다.

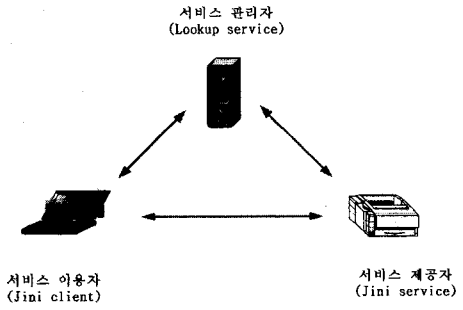


그림 1 Jini 네트워크

2.2 Jini 기술의 하부 구조

Jini 기술은 Jini를 채택한 기기들이 분산되어진 네트워크 환경에서 Jini 연합체를 통해 서비스를 제공하기 위한 하부 구조로 Discovery, Join, Lookup 기능이 있

다. 이는 네트워크상에서 Jini를 채택한 기기들이 간단한 방법으로 서로 연결되어 Jini 연합체에 등록하고, 서로의 자원을 공유하기 위한 수단을 제공하여 준다.

(가) Discovery

Discovery 프로토콜은 서비스 관리자를 찾기 위한 프로토콜이다. 서비스 제공자나 서비스 이용자는 Discovery 프로토콜의 멀티캐스트나 유니캐스트 방법을 이용하여 서비스 관리자에게 요청하고 응답을 받아 서비스 관리자를 찾을 수 있다. 서비스 제공자는 멀티캐스트 메시지를 보내 네트워크 상에서 Lookup 서비스를 찾는다.

(나) Join

서비스 제공자는 Join 프로토콜을 이용하여 서비스 관리자에게 자신의 서비스 객체(Service Object)와 서비스 속성(Service Attribute)을 등록한다. 서비스 제공자가 Discovery를 통해 찾아낸 Lookup 서비스에 클라이언트에게 제공할 서비스 객체와 서비스 속성을 등록하게 된다.

(다) Lookup

서비스 이용자(Client)는 Lookup 프로토콜을 이용하여 서비스 관리자에게 등록되어 있는 서비스 프록시 객체나 속성들을 검색한다. 검색하는 과정에서 원하는 속성에 해당하는 서비스 객체를 가져올 수 있다. Lookup을 통해 얻은 서비스 객체를 이용해 서비스 이용자는 서비스 제공자와 직접 통신하여 서비스 제공자가 제공하는 기능을 이용할 수 있다. 하나의 Jini 네트워크에 여러 개의 Lookup 서비스가 존재할 수 있으며, Lookup 서비스들이 모여서 하나의 Jini 연합체를 형성할 수 있다.

3. Jini 서로게이트 시스템

3.1 Jini 기술 적용시 문제점

소형 장치가 Jini 네트워크에 연결되어 Jini 서비스를 제공하기란 쉽지가 않다. 디지털 온도계와 같은 소형 장치들이 갖는 특징들을 살펴보면 다음과 같다.

- 제한된 시스템 메모리 크기
- 낮은 프로세서 속도
- 제한된 프로그래밍 환경(C/C++ 언어 또는 어셈블리어)
- 다양한 네트워크 계층 존재(Serial, X.10 등등)

소형 장치가 Jini 서비스를 제공하거나 다른 Jini 서비스를 이용하기 위해서는 운영체제 위에 자바가상머신이 탑재되어야 한다. 또한, Discovery와 Join 프로토콜이 RMI를 하부 통신 기반으로 이용하기 때문에 RMI 패키지도 필요하다. 그러나, 장치의 내장형 운영체제 위에 자바가상머신과 RMI 패키지를 탑재하려면 상당량의 메모리와 처리 속도가 빠른 프로세서가 요구된다.

Jini 기술은 자바가상머신 위에서 수행되기 때문에 Jini 서비스를 구현하기 위한 개발 언어로는 자바 언어 밖에 사용할 수 없다. 그러나, 소형 장치들의 개발환경은 C/C++ 언어나 어셈블리어 정도의 개발 환경만 지원하고 있다. 또한, 아직도 대부분의 소형 장치들에 탑재된 프로세서들은 8비트, 16비트가 주류를 이루고 있다. 범용 목적으로 디자인된 일반 PC와는 달리, 장치는 특수 용도에 따라 특정 하드웨어 및 소프트웨어로 제작되어진다. 대부분의 장치들의 통신 환경으로는 RS-232, X.10 등과 같은 non-IP 네트워크 기술이 사용되고 있다. Jini 기술은 IP 네트워크 기반을 전제로 하기 때문에, non-IP 네트워크를 사용하는 장치는 직접 Jini 네트워크에 접속하여 Jini 서비스를 제공하거나 이용할 수가 없다.

이러한 문제점을 해결하기 위해 Jini 개발자 포럼에서는 Jini 네트워크와 Non-Java 장치와의 연결을 위해 표준 인터페이스를 정의하는 연구를 시작하게 되었으며, 현재 IP 네트워크를 지원하는 표준 인터페이스만이 정의된 상태이고, IP 네트워크를 지원하는 서로게이트 호스트를 구현하기 위한 프로젝트도 진행 중이다. 본 논문에서는 Jini 개발자 포럼에서 제시한 IP 표준 인터페이스 API를 지원하는 서로게이트 호스트 및 장치의 Jini 서비스를 직접 구현하여 실험하였다.

3.2 서로게이트 시스템

Jini 네트워크에 직접 접속할 수 없는 장치를 위해 Jini 기능을 대리로 수행해주는 서로게이트 방식을 사용할 수 있다. 서로게이트 시스템은 자바 환경 및 Jini 기능을 탑재하지 못하는 소형 장치를 Jini 네트워크에 연결해주는 역할을 수행하며, Jini 클라이언트에게는 서로게이트 방식으로 연결된 장치인지 Jini 기능이 내장된 장치인지 알 수 없도록 투명성(Trans parence)을 제공해준다.

서로게이트 시스템을 구성하는 중요한 요소는 서로게이트 호스트와 서로게이트 프로토콜이다. 서로게이트 호스트는 Jini 환경이 없는 장치의 서비스 프락시(Service Proxy)를 다운받아 동적으로 실행시켜주는 시스템이다. 서로게이트 프로토콜은 서로게이트 호스트와 장치간의 존재를 찾고 서비스 프락시를 다운받은 후, 서로게이트 호스트 내에 서비스 프락시와 장치의 상태를 유지 관리하기 위해 이용되는 프로토콜이다.

그림 2는 서로게이트 시스템의 구조를 나타낸 것이다. 장치들은 서로게이트 호스트를 통해 Jini 네트워크에 연결되어 서비스를 제공할 수 있다. 장치 1은 장치 2보다

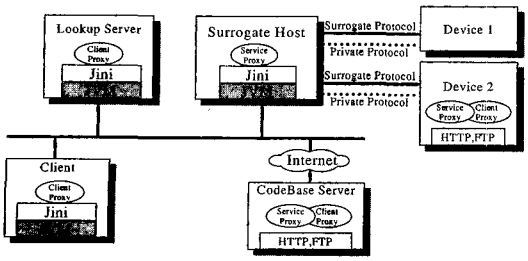


그림 2 서로게이트 시스템 구조

리소스가 더욱 제한되어 서비스 프락시와 클라이언트 프락시도 갖지 못하는 환경이므로 외부 네트워크의 코드베이스(Code Base) 서버에서 다운받도록 프락시 URL을 서로게이트 호스트에게 알려준다. 만약 코드베이스 서버가 장치를 제공한 회사에서 관리될 경우, 향후 서비스의 기능 개선 및 유지 관리의 측면에서 많은 장점을 가질 수 있다. 장치에 서비스 프락시 코드를 내장하면 서로게이트 호스트가 직접 다운받을 수 있는데, 이 경우에는 코드베이스의 다운 방식에 따라 웹서버나 FTP서버가 추가로 필요하게 된다.

서로게이트 호스트는 Interconnect Adapter 모듈, Host Resource Manger 모듈, Surrogate Activator 모듈로 구성된다. 그림 3에서와 같이 Interconnect Adapter 모듈은 장치를 검색하고 서로게이트 프락시를 다운받는 기능을 담당한다. 또한, 다운받은 서로게이트 프락시 코드를 Surrogate Activator 모듈에게 넘겨주는 기능을 담당한다. Host Resource Manager 모듈은 Interconnect Adapter와 서로게이트 호스트 메모리 상에 실행된 서로게이트 서비스 프락시가 시스템의 자원을 사용할 수 있도록 관리하는 모듈이다. 서비스 프락시는 Host Resource Manager가 허용하는 정책 범위 내에서 시스템의 메모리 및 저장장치 등을 사용할 수 있다.

Interconnect Adapter 모듈은 다양한 네트워크를 지원하기 위해 각각의 네트워크마다 대응하는 Inter connect Adapter가 필요하게 된다. 장치와 서로게이트

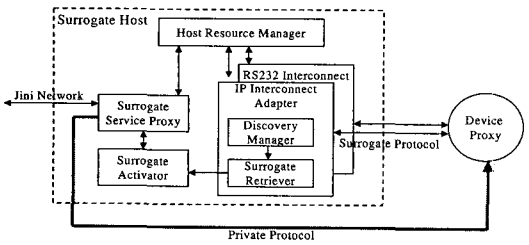


그림 3 서로게이트 호스트 모듈

시스템이 연결되는 망이 IP 네트워크이면 IP Inter connect Adapter 모듈이 필요하게 된다.

Surrogate Activator 모듈은 자바의 동적 클래스 로딩 기능을 이용하여 서로게이트 프락시를 활성화 또는 비 활성화시키는 역할을 담당한다.

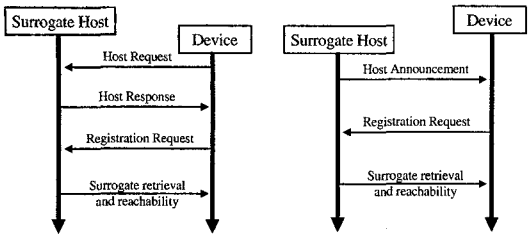
3.3 서로게이트 프로토콜

서로게이트 프로토콜이란 장치가 서로게이트 호스트를 찾거나, 또는 서로게이트 호스트가 장치에게 자신의 존재를 알리고 장치를 서로게이트 호스트에 등록하기 위한 상호 규약이다. 또한 프락시로 사용될 서로게이트 서비스 프락시 코드를 장치에서 다운받는 방식과 서로게이트 호스트와 장치간의 상호 존재 유무를 검사하는 방식을 포함하고 있다.

그림 4는 서로게이트 호스트와 장치간에 서로의 존재를 알리고 등록하는 과정을 나타내고 있다. 그림 4의 (a)는 서로게이트 호스트가 Jini 네트워크에 이미 존재하는 환경에 장치가 나중에 접속한 경우이다. 접속한 장치는 네트워크 상에 자신의 존재를 알리기 위해서 멀티캐스트 메시지를 내보낸다(Host Request Message). 서로게이트 호스트가 장치의 멀티캐스트 메시지를 받으면 호스트의 정보를 포함하는 호스트 응답 메시지(Host Response Message)를 해당 장치에게 보낸다. 장치가 보낸 멀티캐스트 요청 메시지에 대한 호스트 응답 메시지를 받으면 Jini 서비스로 사용될 서로게이트 프락시 코드의 정보를 등록 요구 메시지(Registration Request Message)에 넣어서 서로게이트 호스트에게 보낸 후, 서비스 프락시 코드를 다운받는 과정을 수행한다.

그림 4의 (b)는 서로게이트 호스트가 처음으로 동작을 시작하거나, 서로게이트 시스템이 오류가 생겨 재시작 될 경우에 이루어진 서로게이트 프로토콜 메시지 교환 절차이다. 서로게이트 호스트는 자신의 존재를 알리기 위해 장치들에게 멀티캐스트 방식으로 메시지를 내보낸다(Host Announcement Message). 장치는 멀티캐스트 포트로 서로게이트 호스트의 메시지를 감시하다가 메시지가 발견되면, 서비스 프락시에 관련된 정보를 포함한 등록 요구 메시지를 서로게이트 호스트에게 보낸다. 등록요구 메시지를 받은 서로게이트 호스트는 장치가 보내온 등록요구 메시지에서 서비스 프락시의 위치 정보를 분석하여 서비스 프락시를 다운받는다.

서로게이트 호스트는 서비스 프락시가 실행되는 동안에는 장치와의 연결 상태를 계속 점검해야 한다. 만약 장치가 도중에 네트워크 연결이 단절되거나 장치의 동작이 정지했을 경우, 서로게이트 호스트의 시스템 메모리에서 서비스 프락시를 제거한다.



(a) 장치가 서로게이트 호스트보다 나중에 시작된 경우
(b) 서로게이트 호스트가 장치보다 나중에 시작된 경우

그림 4 서로게이트 프로토콜

다음은 그림 5의 서로게이트 프로토콜 메시지에 대한 설명이다.

(가) 호스트 요청 메시지(Host Request Message)

장치는 서로게이트 호스트를 찾기 위해 멀티캐스트 UDP 메시지를 사용한다. Protocol Version 필드에 서로게이트 프로토콜의 버전 정보가 들어가게 되는데 현재는 '1' 값을 사용한다. Device address 필드와 Port to connect to 필드는 장치 주소와 호스트 응답 메시지에서 사용될 포트 번호로 구성된다.

(나) 호스트 응답 메시지(Host Response Message)

서로게이트 호스트가 호스트 요청 메시지를 보낸 장치에게 그에 대한 응답으로 보내는 유니캐스트 UDP 메시지이다. Host address 필드와 Port to connect to 필드는 장치가 등록요청 할 때 사용되는 호스트의 주소와 포트번호가 들어간다.

(다) 호스트 광고 메시지(Host Announcement Message)

서로게이트 호스트가 자신의 존재를 알리기 위한 멀티캐스트 UDP 메시지이다. 메시지 구조는 호스트 응답 메시지와 같은 메시지 구조를 갖는다. 단지 호스트 응답 메시지는 유니캐스트 메시지인데 비해, 호스트 광고 메시지는 멀티캐스트 메시지라는 점만 다르다.

(라) 등록 요청 메시지(Registration Request Message)

장치가 서로게이트 호스트에게 서로게이트 등록을 위해 필요한 정보를 보내는 메시지로 그림 5의 (c)와 같다. 서로게이트 URL 필드에는 서비스 프락시 파일을 다운받을 수 있는 주소와 다운받을 때 사용될 프로토콜(HTTP, FTP 등)이 들어간다. Jar 파일 포맷으로 압축된 서비스 프락시 파일은 구현 클래스 파일 및 메인 클래스의 정보를 포함하고 있다. Length of Surrogate URL 필드 값이 '0'이면, 메시지의 서로게이트 필드 내부에 서로게이트가 함께 포함되어 전달된다.

서로게이트 메시지의 크기가 작을 경우에는 이와 같은 방법을 사용하면 따로 서로게이트 Jar 파일을 다운받을 필요가 없지만, 컴파일 시 서로게이트가 함께 포함되므로 서비스의 유지보수가 어려워지는 단점이 있다. Length of Surrogate URL 필드 값이 '0'이 아니면 Surrogate URL 필드에 서비스 프락시의 위치 정보(URL)가 포함된다. 서로게이트 호스트는 Surrogate URL 필드에서 URL 정보를 얻어 서비스 프락시를 다운받는다.

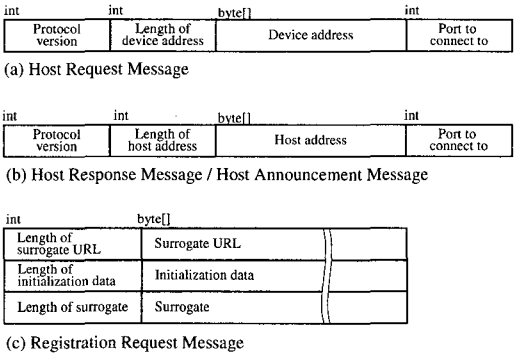


그림 5 서로게이트 프로토콜 메시지 구조

이상의 네 가지 메시지는 서로게이트와 장치가 서로의 존재를 확인하고 서로게이트 서비스를 등록하기 위한 것들이다. 장치와 서로게이트 간의 존재확인(Reachability)을 위한 방법론은 구현 의존적이라고 할 수 있다. 이는 다양한 장치들이 존재하고 각각의 장치마다 다른 특성을 가지고 있으므로 존재확인을 위한 방법도 달라야 하기 때문이다. 예를 들어, 온도를 측정하는 장치는 존재확인 주기가 길어도 되지만, 침입감지를 위한 장치는 존재확인 주기도 짧아야 하며 오류가 발생했을 경우의 대처 방안도 장치마다 달라야 한다. 존재확인 기능을 서로게이트 호스트가 담당할 수 있으며, 또한 장치에서도 담당할 수 있도록 구현할 수 있다.

4. 서로게이트 시스템의 설계 및 구현

4.1 IP Interconnect Adapter

IP Interconnect Adapter는 장치와 서로게이트 프로토콜을 사용하여 서비스 프락시 코드를 다운받는 기능과 장치와의 연결 상태를 관리하는 기능을 담당하는 부분이다. 그림 6과 같이 Process Manager 모듈, Reachability Manager 모듈, Discovery Manager 모듈, Surrogate Retriever 모듈로 구성되어 있다. 각 모듈의 주요 특징들을 살펴보면 다음과 같다.

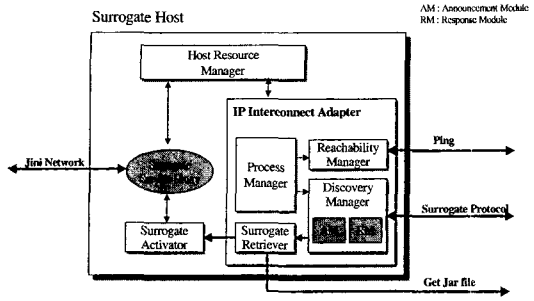


그림 6 서로게이트 호스트

(가) Process Manager 모듈

서로게이트 호스트가 시스템에서 실행될 때 서로게이트 프로토콜 기능을 담당하는 프로세스(Process)를 생성하고 생성된 프로세스를 유지하는 기능을 담당한다. Reachability Manager 모듈과 Discovery Manager 모듈을 각각의 쓰레드로 생성하여 관리한다.

(나) Reachability Manager 모듈

서로게이트 프로토콜을 통해 서로게이트 호스트와 장치간의 존재를 확인한 후, 다운받은 서로게이트 프락시와 장치의 서비스 프락시와의 연결 상태를 관리하는 기능을 담당한다.

연결상태를 관리하는 방법은 2가지 경우가 있다.

- 장치가 연결 상태를 관리하는 방법

장치의 서비스 프락시 내부에 관리 모듈을 갖는 방법으로, 장치마다 다양하게 구현되어질 수 있다.

- 서로게이트 호스트가 별도의 모듈로 연결 상태를 관리하는 방법

다양한 장치 환경에서 공통된 관리 인터페이스를 제공해준다. Discovery Manger가 서로게이트 프로토콜을 통해 알게된 장치의 IP 주소 정보를 사용하여 서로게이트 호스트가 장치에게 Ping 메소드를 주기적으로 호출하여 장치와의 연결 상태를 확인한다.

(다) Discovery Manager 모듈

서로게이트 호스트에서 서로게이트 프로토콜을 이용하여 장치의 존재를 찾거나, 서로게이트 호스트 자신의 존재를 장치들에게 알리는 역할을 담당하는 부분이다. Process Manager에 의해 생성된 Discovery Manger 모듈은 장치와의 서로게이트 프로토콜을 통해 장치의 등록요청메시지 내부의 서로게이트 URL 필드에서 얻은 서비스 프락시의 위치 정보를 Surrogate Retriever 모듈에게 넘겨주는 역할을 수행한다.

(라) Surrogate Retriever 모듈

Discovery Manger 모듈이 제공하는 장치의 서비스 프락시 위치 정보를 가지고 해당 URL주소로부터 다운로드받는 역할을 담당하는 모듈이다. URL의 프로토콜로써 HTTP, FTP, TFTP 등이 사용될 수 있다. 서로게이트 프락시 코드의 위치 정보를 나타내는 형식은 기존 웹 URL과 동일한 형태를 갖고 있다. 본 논문에서는 HTTP URL 주소만 제공하도록 구현하였다.

4.2 Discovery Manager

Discovery Manager는 서로게이트 프로토콜의 역할을 수행하는 모듈이다. 그림 7와 같이 서로게이트 프로토콜을 두 가지의 경우로 분리하여 각 경우에 따라 프로토콜 절차가 다르게 수행된다. 각각의 경우를 담당하는 모듈을 쓰레드로 생성하여 관리하게 된다.

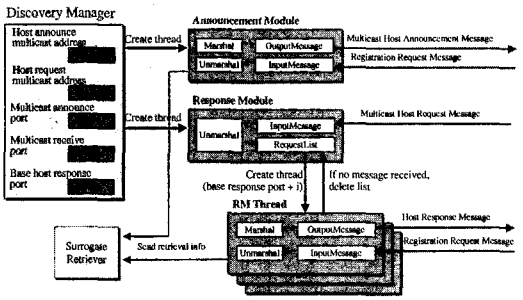


그림 7 Discovery Manager 모듈

서로게이트 프로토콜은 유니캐스트 방식뿐만 아니라 멀티캐스트 전송방식을 사용하고 있다. 서로게이트 시스템의 Discovery Manger 모듈에 사용된 멀티캐스트 주소와 포트는 Jini에서 사용되는 멀티캐스트 주소와 포트를 참고하여 그림 7에서와 같이 설정하고 구현하였다.

그림 4의 (a)는 Response 모듈이 담당하고, 그림 4의 (b)는 Announcement 모듈이 담당하게 된다. 각 모듈의 세부 구현은 다음과 같다.

(가) Response 모듈

장치가 자신의 존재를 알리기 위해 호스트 요청 메시지를 멀티캐스트로 전송하게 되는데, Response 모듈에서 멀티캐스트 응답 포트로 멀티캐스트 호스트 요청 메시지를 리스닝하게 된다. 그림 7과 같이 리스닝 포트로 들어오는 메시지를 분석하여(Unmarshal 과정), 요청 리스트를 관리하는 RequestList가 각각의 요청 메시지마다 해당 장치에게 유니캐스트 호스트 응답 메시지를 보내는 쓰레드를 생성하고 관리한다. 생성된 쓰레드는 멀티캐스트 호스트 요청 메시지의 Device Address 필드에서 얻어진 장치의 주소와 쓰레드가 생성될 때 자동으로 할당된 포트를 호스트 응답 메시지에 담아서 장치에

게 보낸 후 등록 요청 메시지를 기다린다. 호스트 응답 메시지에 대한 응답으로 등록 요청 메시지가 도착하면 메시지 필드를 분석하여(Unmarshal 과정), 서로게이트 프락시의 URL 주소를 얻은 후 이 정보를 가지고 Surrogate Retriever 모듈을 호출한다.

(나) Announcement 모듈

서로게이트 호스트가 처음 실행되거나, 실행 중 문제가 발생하여 재시작 되었을 때 Discovery Manager는 Announcement 모듈을 쓰레드로 실행한다. Host announce multicast address(224.0.1.56)와 Multicast announce port(4260)로 멀티캐스트 호스트 광고 메시지를 보내게 된다. 멀티캐스트 호스트 광고 메시지를 받은 장치가 이에 대한 응답으로 보낸 등록 요청 메시지를 받아서 장치의 서비스 프락시 위치정보를 얻는다. 이 정보를 매개변수 값으로 Discovery Manager 모듈은 Surrogate Retriever 모듈을 실행한다.

4.3 Surrogate Retriever

Discovery Manager가 서로게이트 프로토콜을 통해 얻은 서비스 프락시의 위치정보를 바탕으로 Surrogate Retriever는 해당 주소에서 서비스 프락시를 서로게이트 호스트로 다운받는 기능을 수행한다. 그림 8과 같이 Surrogate Retriever는 Jar 파일을 다운받는 Save Jar File 모듈, 자바 보완 정책을 담당하는 Policy Manager 모듈, Jar 파일을 메모리에 올리는 Jar Class Loading 모듈로 구성되어 있다.

다운받은 Jar 파일은 서비스 프락시의 정보가 포함된 MENIFEST 파일에서 서비스 프락시의 실행 클래스 이름을 얻어 후, 이 정보를 매개 변수로하여 자바가상머신이 제공하는 동적 클래스 로딩(Dynamic Class Loading) 기능을 이용해 서비스 프락시를 서로게이트 호스트에서 동적으로 실행시킨다. 서비스 프락시 코드는 Lookup 서버를 찾아 서비스를 등록하는 부분을 담당하므로 서로게이트 호스트가 서비스 프락시를 Lookup 서버에 등록하는 부분은 담당하지 않는다. 서비스 프락시가 Lookup 서버에 등록하는 과정은 장치의 서비스 프락시 코드를 구현하는 사람이 구현해야 한다.

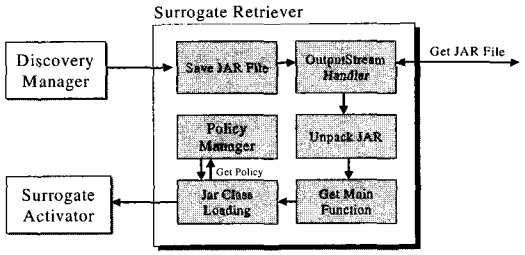
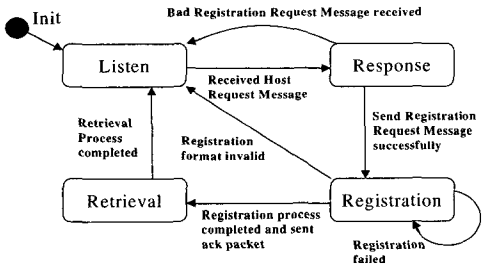


그림 8 Surrogate Retriever 모듈

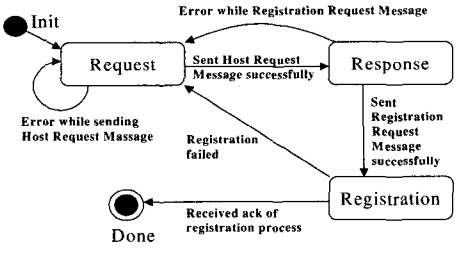
4.4 서로게이트 프로토콜 상태천이도

그림 9는 서로게이트 프로토콜의 메시지 교환 절차를 상태천이도로 나타낸 것이다. 그림 4의 (a)의 서로게이트 호스트와 장치간의 메시지에 흐름에 따른 상태천이도를 나타내었다. 그림 9의 (a)는 서로게이트 호스트가 Listen 상태로 대기하다가 장치가 Host Request를 요청하는 Response 상태로 천이한다. 장치에게 Response 메시지를 보내고 Registration 상태로 천이한다. 만약 Response 메시지 보내기에 실패하면 다시 Listen상태로 되돌아간다. Registration 상태에서 등록과정이 실패하면 같은 과정을 3회 반복한 후, Listen 상태로 복귀한다. Retrieval 상태에서 서로게이트 호스트는 서비스 프락시 코드를 다운받은 후, 다시 Listen 상태로 천이하여 새로운 장치의 요청을 기다린다.

그림 9의 (b)는 장치가 서로게이트 호스트에게 호스트 요청 메시지를 보낸 후, Response 상태로 천이하게 된다. 서로게이트 호스트로부터 응답 메시지가 오지 않으면 다시 호스트 요청 메시지를 보내게 된다. 서로게이트 호스트로부터 응답메시지를 받으면 등록 메시지를 보내



(a) 서로게이트 호스트의 상태천이도



(b) 장치의 상태천이도

그림 9 서로게이트 프로토콜의 상태천이도

고 서로게이트 프로토콜 과정을 종료하게 된다.

4.5 서로게이트 시스템 구현

(가) 구현 환경

서로게이트 시스템의 구현을 위해 사용된 운영체제는 i386용 RedHat Linux 6.2이다. 구현한 서로게이트 시스템은 자바가상머신 위에서 동작하므로 운영체제에 독립적인 특성을 갖는다. 서로게이트 시스템의 구현을 위한 프로그래밍 개발 환경은 JDK(Java Development Kit) 1.2.2와 Jini Development Kit 1.01을 사용하여 순수 자바 언어로 구현하였다.

(나) 서로게이트 시스템 클래스 구현

그림 10은 자바로 구현한 서로게이트 호스트의 클래스 다이어그램이다. 서로게이트 표준 문서에서 제공하는 인터페이스를 통해 서로게이트 호스트와 서비스 프락시 간에 정보를 교환하게 된다. IP Interconnect Adapter 클래스는 서로게이트 표준에서 제공하는 IPInterconnect Context 인터페이스를 구현한 것이다. Surrogate Activator 인터페이스를 구현한 SurrogateProxy Manager는 서로게이트 시스템의 메모리 내에서 동적으로 실행되는 서로게이트 프락시 코드의 생명주기를 관리하는 클래스이다.

5. 시험 및 결과

5.1 시험 환경

구현한 서로게이트 시스템을 시험하기 위한 환경은 그림 11과 같이 4대의 컴퓨터를 네트워크로 연결하였다. 서로게이트 호스트는 Redhat Linux 6.2 운영체제를 사용하는 펜티엄 컴퓨터에 설치하였으며, JDK 1.2.2 및 Jini 1.01 환경을 구성하였다. 장치는 Alpha Station에 운영체제로 Redhat Linux 6.0을 이용하였다. Jini 실행 환경이 다양한 OS에서 동작할 수 있음을 보여주기 위해 다양한 OS 및 CPU를 사용하는 시스템을 사용하여 구성하였다. 장치가 제공하는 서비스 프로그램 및 서로게이트 프로토콜은 C 언어를 사용하여 구현하였다. 내장형 장치는 PC와 다른 CPU 및 OS가 사용되거나 OS조차 없는 Monitor 환경 정도가 대부분이다. 본 논문에서는 Alpha 칩과 Linux OS를 사용하는 시스템을 내장형 장치로 가정하고 구현하였다. 구현된 서로게이트 시스템과 통신하는 장치의 OS나 CPU가 바뀌더라도 동일한 프로토콜로 통신하므로 서로게이트 호스트 부분은 변경사항이 없다. 단, 각 장치가 서로게이트 호스트와 통신하기 위해 짜여진 장치 부분의 코드는 장치의 OS나 CPU마다 다를 수 있다.

서비스 프락시 코드를 다운받기 위한 코드베이스 서버는 외부 서브넷 망에 존재하는 Solaris 운영체제 기반의 웹서버를 이용하였다. 클라이언트는 Windows NT 운영체제를 사용하는 펜티엄 컴퓨터를 사용하였으며,

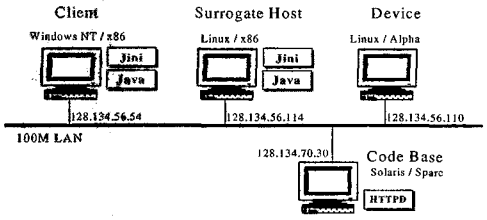


그림 11 실험 환경

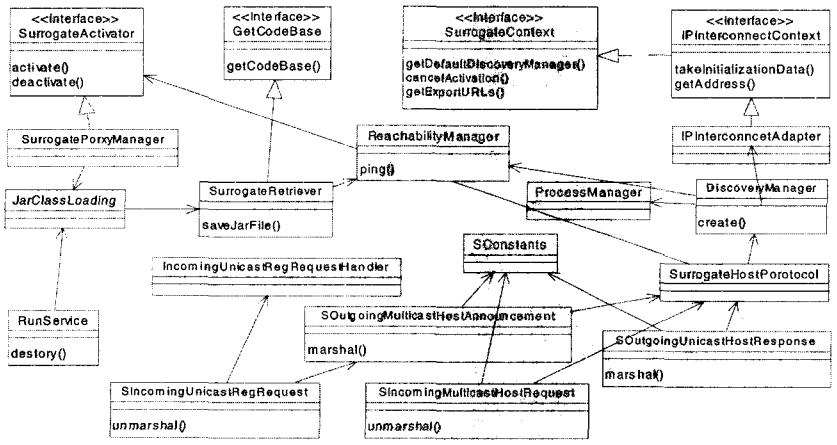


그림 10 서로게이트 호스트의 클래스 다이어그램

JDK 1.2.2 및 Jini 1.01을 사용하여 자바 언어로 Jini 클라이언트 프로그램을 작성하였다.

5.2 시험 시나리오

본 논문에서 구현한 서로게이트 시스템이 정상적으로 동작하는지를 확인하기 위하여 Jini 예제 프로그램을 작성하여 시험하였다. Jini 클라이언트가 Lookup 서버에서 장치의 현재 시각을 알려주는 Jini 서비스를 찾은 후, 장치 현재 시각을 얻는 시나리오를 구현한 예제를 사용하였다. 그림 12는 Jini 클라이언트가 장치가 제공하는 Jini 서비스를 이용하기 위한 동작 과정의 메시지 흐름을 나타낸 그림이다. Jini 클라이언트는 Lookup 서버에 원하는 서비스를 등록하여 해당 서비스가 등록되면 이벤트로 알려주는 방식으로 프로그램을 작성하였다. 장치는 서로게이트 프로토콜을 사용하여 서로게이트 호스트를 찾는다.

장치가 서로게이트 호스트에게 멀티캐스트로 호스트 요청 메시지를 보내면(1), 서로게이트 호스트는 장치에게 호스트 응답 메시지로 응답한다(2). 장치는 등록 요청 메시지로 서비스 프락시를 서로게이트 호스트에 등록한다(3). 서로게이트 호스트는 서로게이트 프로토콜을 통해 얻은 서비스 프락시의 URL에서 서비스 프락시 코드를 다운받아 서로게이트 호스트 메모리에서 실행시킨다(5). 실행된 서비스 프락시는 Lookup 서버를 찾아 Jini 서비스를 등록시키며(6), Lookup 서버는 이벤트를 등록한 Jini 클라이언트에게 이벤트 메시지를 보내게 된다(7). 이벤트를 받은 Jini 클라이언트는 Jini 서비스의 프락시 코드를 다운받아 서로게이트 호스트에 존재하는 서비스 프락시가 제공하는 함수를 호출하게 된다(10). Jini 클라이언트가 GetTime함수를 서비스 프락시에게 호출하면(11) 서비스 프락시는 실제로 장치에게 get Time함수를 호출하여(12) 장치의 현재 시각을 얻어낸 후(13) Jini 클라이언트에게 되돌려 준다(14).

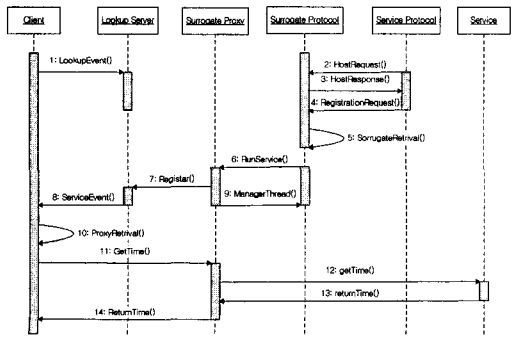


그림 12 실험 예제의 메시지 호출 과정

이러한 과정을 통해 Jini 클라이언트에게는 장치가 Jini 네트워크에서 Jini 서비스를 제공하는 하나의 서비스 제공자로 보이게 되는 것이다. 서로게이트 호스트는 Jini 네트워크에 직접 연결되어 서비스를 제공하지 못하는 장치들을 다른 Jini 클라이언트들에게 일반적인 Jini 서비스처럼 보이도록 해주는 중개자 역할을 담당한다. Jini 클라이언트는 서로게이트 호스트를 통해서 장치의 현재시각을 얻을 수 있다.

5.3 시험 결과

실험에 사용된 예제는 일반적인 Jini 네트워크 환경에서 사용되는 것으로, Jini 서비스 제공자의 현재 시각을 얻는 Jini 프로그램이다. Jini 클라이언트 쪽의 예제는 그대로 사용하였으며, 장치에서 다운받아 서로게이트 호스트 내에서 동작하는 프락시 코드는 기존 Jini 서비스 예제에 장치와 통신하는 부분을 추가하여 실험하였다. 그림 13은 Jini 클라이언트가 서로게이트 호스트를 통해 장치의 현재시각을 얻어 화면에 출력하는 과정을 나타낸 것이다. 장치의 시스템 시각을 얻어와서 클라이언트의 출력 창에 장치의 시각을 표시하고 있다. Jini 클라이언트는 일반적인 Jini 서비스 제공자와 함께 사용되는 예제를 그대로 사용하였으므로, 서로게이트 호스트를 통해 장치의 서비스를 이용할 수 있다면 기존 Jini 서비스 제공자와 상호연동성이 보장됨을 알 수 있다.

기존 Jini 클라이언트는 서로게이트 호스트를 통해서 non-Java 장치가 제공하는 서비스를 기존 Jini 서비스로 인식하고 서비스를 이용하였다. Jini 클라이언트는 서로게이트 호스트를 통해 장치가 제공하는 Jini 서비스인지 아니면 일반적인 Jini 서비스 제공자가 제공하는 서비스인지 알 수 없는 투명성을 제공받는다. 이상과 같은 실험 과정을 통해 자바 환경을 갖출 수 없는 장치가 본 논문에서 구현한 서로게이트 시스템을 이용하여 Jini 네트워크에 접속된 Jini 클라이언트들에게 Jini 서비스를 제공할 수 있음을 확인하였다.

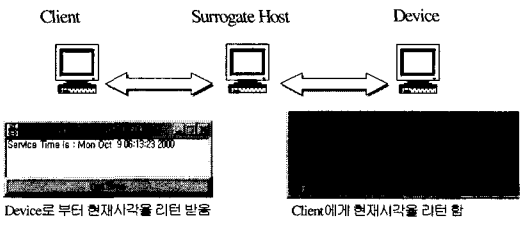


그림 13 장치의 현재시각을 얻는 실험

6. 결 론

분산 네트워크 미들웨어인 Jini 기술은 자바 플랫폼 위에서 동작하는 환경이기 때문에, 자바 환경을 갖출 수 없는 소형 장치는 Jini 네트워크에 연결되어 서비스를 제공할 수 없다. 이러한 문제점을 해결하기 위해 장치가 제공하지 못하는 Jini 기술의 Lookup 및 Join 기능을 대신하는 서로게이트 방식이 제안되었다.

본 논문에서는 non-Java 장치가 Jini 네트워크에 접속하여 서비스할 수 있는 서로게이트 시스템을 설계하고, 장치와 서로게이트 시스템이 동적으로 서로의 존재를 확인하고 등록한 후 서로간의 연결을 확인하는 서로게이트 프로토콜을 설계 및 구현하였다.

서로게이트 시스템을 이용해 non-Java 장치도 Jini 네트워크에 접속하여 서비스를 제공할 수 있다. 서로게이트 시스템은 아직 표준화 작업중이며, 구체적인 설계 및 구현이 이루어지지 않고 있다. 현재 서로게이트 호스트와 장치간의 서로게이트 프로토콜과 기본적인 인터페이스만 정의되어 있다. 서로게이트 표준 문서에서 정의하고 있는 서로게이트 프로토콜과 서로게이트의 기능을 만족하기 위해, 서로게이트 프로토콜과 Inter connect Adapter를 설계하고 리눅스 시스템에서 구현하였다. 구현한 서로게이트 시스템은 순수 자바로 구현하였기 때문에 자바가상머신을 갖춘 다양한 시스템에서 적용할 수 있다.

구현한 서로게이트 시스템이 정상적으로 동작하는지 실험 환경을 구성하여 Jini 서비스 실험을 하였다. 실험에서 Jini 클라이언트는 서로게이트 호스트를 통해 non-Java 장치의 서비스를 일반 Jini 서비스처럼 인식하고 장치가 제공하는 서비스를 이용할 수 있음을 확인하였다.

서로게이트 호스트는 다양한 네트워크에 존재하는 장치들을 지원하기 위한 시스템이다. 현재 IP 망을 위한 IP Interconnect Adapter 부분만 표준화 작업중이며, 향후 다양한 네트워크를 지원하기 위한 Interconnect Adapter가 필요하게될 것이다. 따라서, 향후 연구과제로는 다양한 네트워크에서의 적용하기 위해 non-IP 망에서의 Interconnect Adapter 설계 및 구현에 관한 연구가 필요하다.

참 고 문 헌

[1] OMG, "The Common Object Request Broker: Architecture and Specification, Revision 2.3.1," Object Management Group Inc., 1999.

[2] G. Eddon and H. Eddon, Inside Distributed COM, Microsoft Press, 1998.

[3] Sun Microsystems Inc., Jini Connection Technology, http://www.sun.com/jini/

[4] Sun Microsystems Inc., "Jini Architecture Specification," 1999.

[5] Sun Microsystems Inc., "Jini Device Architecture Specification," 1999.

[6] C. McDowell and K. Shankari, "Connecting Non-Java Devices to a Jini Network," Proc. of 33rd International Conference on Technology of Object-Oriented Languages, pp. 45-56, June 2000.

[7] P. Perrone and V. Chaganti, "Jini in the Box," Embedded Systems Programming, Vol. 12, pp. 55~64, November 1999.

[8] W. Edward, *Core Jini*, Prentice Hall, 1999.

[9] S. Oaks and H. Wong, *Jini in a Nutshell*, O'reilly, March 2000.

[10] Jini Community, "Jini Surrogate Project," http://developer.jini.org/exchange/projects/surrogate/

[11] Jini Community, "Jini Technology Surrogate Architecture Specification, Ver. 0.5," 2000.

[12] Jini Community, "Jini Technology IP Interconnect Specification, Ver. 0.3," 2000.

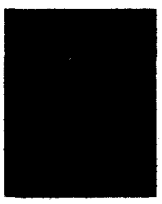
[13] S. Li, *Professional Jini*, Wrox Press, 2000.

[14] D. Ayers and S. Li, *Professional Java Server Programming*, Wrox Press, 1999.

[15] B. McCarty and L. Cassady-Dorion, *Java Server Programming*, Wrox Press, 2000.

[16] M. Hughes, M. Shoffner and D. Hamner, *Java Network Programming*, 2nd Ed., Mannig Publications Co., 1999.

[17] 최현석, 모상덕, 정광수, 이혁준, "Non-Java 장치를 지원하기 위한 Jini 서로게이트 시스템 설계", 한국정보과학회 추계 학술발표논문집 Vol. 27, No. 2, pp. 304-306, 2000.



최 현 석

1999년 광운대학교 전자통신공학과 학사
2001년 광운대학교 전자통신공학과 석사
2001년 ~ 현재 삼성전자 근무. 관심분야는 컴퓨터통신, 인터넷 프로토콜, 내장형 시스템

모 상 덕

1998년 광운대학교 전자통신공학과 학사
2000년 광운대학교 전자통신공학과 석사
2000년 ~ 현재 광운대학교 전자통신공학과 박사과정. 관심분야는 분산처리, 인터넷 QoS, 내장형 시스템

정 광 수

정보과학회논문지 : 정보통신
제 29 권 제 3 호 참조

오 승 준

1980년 2월 서울대학교 전자공학과 졸업(학사). 1982년 2월 서울대학교 전자공학과 대학원 졸업(석사). 1988년 5월 미국 Syracuse University 전기 및 컴퓨터공학과 졸업(박사). 1982년 3월 ~ 1992년 8월 한국전자통신연구원 근무(멀티미디어연구실 실장). 1986년 7월 ~ 1986년 8월 NSF Supercomputer Center 초청 학생연구원. 1987년 5월 ~ 1988년 5월 Northeast Parallel Architecture Center 학생연구원. 1992년 9월 ~ 현재 광운대학교 전자공학부 및 정보통신연구원 교수 (멀티미디어연구실). 2000년 3월 ~ 현재 (주)인티스 정보통신연구소 연구소장. 관심분야는 비디오처리, 비디오 및 영상압축, 멀티미디어시스템