DOI: 10.3745/KIPSTA.2009.16-A.4.227

분산되어 있는 OSGi 프레임워크에서 효과적인 서비스 공유 방안

윤 기 현[†] · 김 은 회[††] · 최 재 영[†††]

요 약

홈 네트워크 분야에서 주로 적용되었던 OSGi 기술은 현재 다양한 도메인의 이질적인 장치에 탑재되고 있다. 분산되어 있는 OSGi 프레임워크 환경에서 효과적인 서비스를 제공하기 위해서는 프레임워크들간의 상호작용이 필수적이다. 하지만 현재까지 공개된 OSGi 스펙에서는 원격 OSGi 프레임워크에 등록된 서비스를 공유할 수 있는 방안을 제공하지 않고 있다. 이와 같은 문제점을 해결하기 위하여 기존의 분산 미들웨어 기술인 JXTA 또는 Web Services를 활용하는 기술들이 연구되었으나, 이러한 기술들은 컴퓨팅 자원을 많이 소모하고 또한 부가적인 서비스 변환과정이 필요하다는 문제점이 있다. R-OSGi는 OSGi 기술만을 사용하여 원격 서비스를 공유할 수 있는 방안을 제공하지만, 중앙집중적인 구조를 가짐으로써 병목현상이나 SPOF (Single Point Of failure)가 발생할 수 있다. 본 논문에서는 P2P 기반의 효과적인 서비스 공유 방안인 RSP (Remote Service Provider)를 제안한다. RSP는 OSGi 자체 기술만을 사용하므로 서비스를 공유하기 위하여 추가적으로 다른 소프트웨어를 설치하거나, 그 소프트웨어를 사용하기 위한 변환과정이 필요 없다. 또한 P2P 방식의 서비스 발견 메커니즘을 사용하여 병목현상을 해결하고 확장성을 높일 수 있다. 그리고 RSP는 원격 서비스에 대한 투명성을 제공하고, 원격 서비스의 상태 변화를 즉각 통보하여 원격 서비스의 신뢰성을 보장하는 특징을 가진다.

키워드 : OSGi, 서비스 공유, 서비스 디스커버리, P2P

An Effective Service Sharing Scheme on Distributed OSGi Frameworks

Kihyun Yun[†] · Eunhoe Kim[††] · Jaeyoung Choi[†††]

ABSTRACT

OSGi technology has applied to Home Network, but now it is loaded into even heterogeneous devices in various domains. Therefore, it is necessary to cooperate with each other framework for offering effective services in distributed OSGi frameworks. However until now, OSGi specification doesn't provide any methodthat can share the services registered on remote OSGi frameworks. In order to solve this problem, there have been several researches that used existing distributed middleware technologies such as JXTA and Web Services. However these technologies have some weakness, that is, they consume lots of computing resources and need additional process to transform the services. A middleware called R-OSGi uses only OSGi technology for sharing remote OSGi services, but R-OSGi may have a communication bottleneck and SPOF (Single Point of Failure) problem, because it has a central service registry. In this paper we present RSP (Remote Service Provider), which is a P2P-based effective service sharing scheme on distributed OSGi framework. RSP doesn't need to install additional software nor have the additional transformation process of the service representation, because it uses only OSGi technology. In addition it doesn't have any bottleneck problem and improves scalability by providing the service discovery mechanism using P2P. RSP can also access remote services transparently and it can guarantee reliability by sending an immediate notice about changes of the remote services.

Keywords : OSGi, Service Sharing, Service Discovery, P2P

1. 서 론

인터넷의 폭발적인 성장은 IT를 비롯한 과학기술 분야뿐만 아니라 우리 생활에도 많은 변화를 주었다. 이러한 변화 가운데 새롭게 등장한 분야는 홈 네트워크이다. 또한 유무선 기술과 임베디드 기술의 발전으로 사용자는 홈 네트워크

※ 이 논문은 2006년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2006-005-J03803).
† 준 회 원 : 숭실대학교 컴퓨터학과 석사과정
†† 정 회 원 : 숭실대학교 정보미디어기술연구소 전임연구원
††† 종신회원 : 숭실대학교 컴퓨터학부 교수
논문접수 : 2009년 2월 12일
수 정 일 : 1차 2009년 5월 25일
심사완료 : 2009년 6월 5일

와 인터넷을 연동하여 가정내의 정보 가전 기기를 언제 어디서든지 자유롭게 사용하기를 원하게 되었다. 이러한 유비쿼터스 홈 네트워크 환경을 제공하기 위해서는 가정 내의 모든 기기들을 하나의 정보 시스템에 통합하고 제어하는 미들웨어가 반드시 필요하다. 지금까지 홈 네트워크를 지원하기 위하여 UPnP, Jini, Havi 등과 같은 다양한 미들웨어 기술들이 개발되었다. 그러나 홈 네트워크 분야에 IT 기업뿐만 아니라 통신, 가전, 건설 등과 같은 다양한 업체들이 참여하게 되면서 각 업체에서 제공하는 이질적인 서비스들을 상호 운용할 수 있는 새로운 미들웨어 기술이 필요하게 되었다. OSGi는 이러한 요구를 충족시키기 위해 개발된 기술 표준이며, 현재 IBM, 삼성전자, 모토로라, NTT, 오라클, 썬 마이크로시스템즈, 노키아 등 세계 유수의 기업들이 컨소시엄을 구성하여 표준화 작업을 진행하고 있다 [1].

초기 OSGi 기술은 홈 네트워크 분야에 초점을 맞추어 설계되었다. 홈 네트워크는 OSGi 기반의 홈 게이트웨이 한 대를 중심으로 가정내의 다양한 장치들을 연결하는 기본 구조를 가진다 [2, 3, 4, 5]. CES (Consumer Electronics Show) 2004에서는 OSGi 프레임워크 기반의 정보 가전 기기를 홈 게이트웨이로 구축하고, 홈 게이트웨이에 연결된 서로 다른 정보 가전 기기들을 OSGi 프레임워크를 사용하여 상호 운용할 수 있다는 것을 보여주었다 [6]. 한편, 유비쿼터스 컴퓨팅 기술이 가정뿐만 아니라, 통신, 자동차, 공장, 사무실 등으로 도메인이 다양화됨에 따라, OSGi 기술 또한 핸드폰, 자동차, 데스크탑 PC, 엔터프라이즈 등과 같은 다양한 영역에서 적용되고 있는 추세이다. 따라서 OSGi Alliance에서는 CPEG (Core Platform Expert Group), VEG (Vehicle Expert Group), MEG (Mobile Expert Group), EEG (Enterprise Expert Group)와 같은 전문 그룹을 구성하여 OSGi 기술을 각 전문 분야에 적용하기 위한 활발한 연구를 수행하고 있다 [7].

또한 OSGi 프레임워크는 자바 기반의 여러 컴포넌트들을 효율적으로 서로 함께 운용할 수 있는 소형 레이어에 해당하는 기술로, 한 대의 자바 가상 머쉰이 운용되는 어떤 장치에도 탑재될 수 있다는 특징이 있다. 따라서 OSGi 플랫폼은 가상 자바 머쉰이 동작할 수 있는 크고 작은 다양한 성능의 장치에 탑재되어 유비쿼터스 서비스를 제공하기 위한 미들웨어로서 역할을 담당하는 추세이다. 따라서 한 대의 OSGi 프레임워크가 게이트웨이 역할을 하던 기존 홈 네트워크와는 달리, 사용자에게 알맞은 서비스를 제공하기 위하여 여러 대의 분산된 OSGi 프레임워크들이 서로 상호작용해야 할 필요성이 증대되고 있다.

분산된 OSGi 프레임워크들이 상호작용하기 위해서는 먼저 OSGi 프레임워크간에 서비스 공유가 이루어져야 한다. 그러나 현재까지 나온 OSGi 스펙에서는 원격 OSGi 프레임워크에 등록된 서비스를 공유할 수 있는 방법을 제공하지 않고 있다. 원격 OSGi 서비스를 공유하기 위한 관련 연구로는 기존의 분산 기술인 P2P 기반의 JXTA, Web Services, SLP (Service Location Protocol)와 같은 기술들을 활용하는 방안들이 연구되었다. 그러나 이러한 연구들은 OSGi 프레임워크에 JXTA 또는 Web Services를 번들로 실행시키거나, 부가적인 서비스 변환과정이 필요하며, 또한 SLP를 사용했을 때는 프로토콜의 특성상 OSGi 내부의 서비스 레지스트리와는 별도로 중앙 서비스 레지스트리를 유지해야 한다는 문제점이 있다.

따라서 본 논문에서는 분산 OSGi 프레임워크의 서비스 공유를 위한 RSP (Remote Service Provider)를 제안한다. RSP는 OSGi 자체 기술만을 사용하며, 원격 OSGi 서비스를 로컬 OSGi 서비스처럼 호출하여 사용할 수 있는 서비스 투명성을 제공한다. 또한 OSGi 번들로 구동되어 작은 장치에도 실행될 정도로 가벼우며, 원격 서비스 발견을 위해 P2P 방식을 사용함으로써 서버 병목현상이 발생하지 않는다는 특징이 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 논하고, 3장에서는 OSGi 프레임워크의 기본 구조를 설명한다. 4장에서는 RSP의 설계와 구현에 대해서 논하고, 5장에서는 RSP의 성능을 평가한다. 마지막으로 6장에서 본 논문의 결론과 향후 연구 과제를 제안한다.

2. 관련 연구

OSGi 릴리즈 3 스펙에서는 OSGi 프레임워크를 사용하여 네트워크로 연결되어 있는 원격 장치를 제어하기 위한 방법으로, OSGi Jini와 OSGi UPnP 서비스를 정의하고 있다. OSGi Jini와 OSGi UPnP 서비스 스펙은 각각 OSGi 서비스와 Jini 및 UPnP 서비스와의 상호연동을 지원한다. 그러나 OSGi Jini 서비스는 Jini 디바이스를 완벽하게 제어할 수 없다는 문제가 있으므로, OSGi 릴리즈 4 스펙에서는 현재 OSGi UPnP 서비스 스펙만을 지원하고 있다 [9]. 또한 현재까지 공개된 OSGi 스펙에서는 UPnP 서비스와의 상호연동을 위한 OSGi 서비스 스펙을 제공하고 있지만, 분산된 OSGi 프레임워크에서 동작하고 있는 OSGi 서비스들의 상호운용을 위한 서비스 공유방안은 지원하지 못하고 있는 실정이다. 그러므로 분산 OSGi 프레임워크들이 실행하고 있는 서비스들을 서로 공유하여 상호운용할 수 있도록 하기 위한 연구들이 진행되고 있다. 대표적인 연구들로는 JXTA를 이용한 P2Pcomp, Web Services 기술을 이용한 스마트 아키텍쳐, 그리고 SLP을 이용한 R-OSGi 등의 연구들이 있다. 각 연구들을 간략하게 살펴보도록 한다.

2.1 JXTA를 이용한 P2Pcomp

컴포넌트를 기반으로 하는 분산 어플리케이션을 보다 용이하게 구현하려는 목적으로 개발된 P2Pcomp는 OSGi 프레임워크와 자바 기반의 P2P 통신 프레임워크를 제공하는 JXTA (Juxtapose) [10]를 사용하여 설계된 시스템이다. 로컬 OSGi 프레임워크는 원격의 OSGi 프레임워크의 서비스를 호출하기 위해 원격 OSGi 프레임워크와 통신 채널을 가져야 하는데, P2Pcomp에서는 JXTA를 사용하여 통신 채널

을 제공한다. JXTA 기능을 PortManager라 불리는 OSGi 번들로 구현하고, OSGi 프레임워크에 탑재하여 원격 OSGi 프레임워크의 서비스들을 P2P 방식으로 공유한다. 원격 OSGi 프레임워크의 서비스를 공유하기 위해 JXTA를 사용함으로써 개발자는 서비스를 공유하기 위한 하위 레벨을 신경쓰지 않고, 서비스 로직을 구현하는데 집중할 수 있다 [11]. 그러나 원격의 OSGi 프레임워크와 통신하기 위하여 OSGi 프레임워크에는 JXTA의 모든 기능이 구현된 PortManager라는 번들을 설치해야 하는 부담이 발생한다. 또한 OSGi 프레임워크는 컴퓨팅 자원이 부족한 장비에도 많이 탑재되기 때문에 P2P 통신 프레임워크인 JXTA를 구현한 PortManager 번들을 설치할 때 컴퓨팅 성능의 문제가 발생할 수 있다.

2.2 Web Services 기술을 이용한 스마트 아키텍쳐

스마트 아키텍쳐는 OSGi 기반의 장치들로 구성된 스마트 홈에서 이동 에이전트의 마이그레이션을 제공하기 위한 구조를 제안한다. 이동 에이전트의 계획, 성질, 상태 등은 XML 기반 MASML (the Mobile Agent Specification Markup Language)로 기술하고, MASML로 기술된 이동 에이전트는 Web Services를 통하여 OSGi 프레임워크들 사이를 이동하며 실행된다.

스마트 아키텍쳐를 구성하는 각각의 OSGi 프레임워크는 Web Services를 지원하는 번들을 구동하며, MASML 문서를 해석하여 에이전트의 작업을 수행하는 번들을 실행하고 있다. OSGi 프레임워크는 실행 결과를 MASML 문서로 작성하고, Web Services를 통하여 서비스 실행을 요청한 장치 또는 다음 작업을 수행할 장치로 전송한다 [12].

이동 에이전트는 MASML로 기술되고 Web Services를 통하여 원격 OSGi 프레임워크에 전달되어 실행되므로, MASML 문서를 OSGi 서비스 형태로 변환하는 과정이 부가적으로 필요하다. 또한 스마트 아키텍쳐는 원격의 OSGi 프레임워크에서 동적으로 변하는 원격 서비스의 상태 변화를 반영하지 못하기 때문에 원격 서비스에 대한 신뢰성을 제공하지 못한다는 단점이 있다.

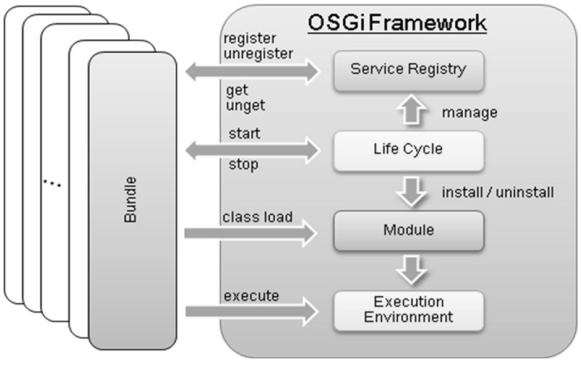
2.3 SLP을 이용한 R-OSGi

R-OSGi는 로컬 OSGi 프레임워크의 서비스를 원격 OSGi 프레임워크에서 접근하여 사용할 수 있도록 지원하는 미들웨어이다. R-OSGi를 구동하는 OSGi 프레임워크는 원격 서비스를 공유하기 위해 SLP (Service locator protocol) [13, 14] 라는 프로토콜을 사용한다. SLP는 중앙 집중적인 서비스 레지스트리를 사용하는 프로토콜이므로 R-OSGi를 구동하는 OSGi 프레임워크들 중 하나는 SLP 서비스 레지스트리를 운영하여 서비스 브로커의 역할을 수행해야 한다. 그러므로 R-OSGi를 사용하는 모든 OSGi 프레임워크는 원격 서비스를 공유하고 사용하기 위해 반드시 한 대의 서비스 브로커와 통신해야 하므로 네트워크의 병목현상과 SPOF (Single Point of Failure) 문제가 발생할 수 있다. 이러한 단점 때문에 R-OSGi는 서비스 브로커를 발견하지 못했을 경우, 다시 멀티캐스트를 사용하여 다른 원격 서비스를 발견하게 된다. 반면에 R-OSGi는 정적인 결과 값을 가지는 원격 서비스를 로컬 OSGi 프레임워크에 직접 생성하여 원격이 아닌 로컬에서 서비스를 실행하기 때문에, 정적인 결과값을 가지는 원격 서비스 요청시 빠른 실행 결과를 얻는 장점을 가진다 [15].

3. OSGi 프레임워크의 구조

OSGi 핵심 기술은 OSGi에서 어플리케이션에 해당하는 번들의 실행 환경을 제공하는 OSGi 프레임워크이다. 하나의 번들은 한 개 이상의 서비스들로 구성된다. OSGi 프레임워크는 자바 가상 머쉰을 기반으로 작동되고 다수의 번들을 수행하기 위한 번들간의 공유 및 협력을 제공할 수 있다. 이를 위해 OSGi 프레임워크는 모듈화, 라이프 사이클 관리, 서비스 레지스트리 기능을 가지고 있다 [7, 8].

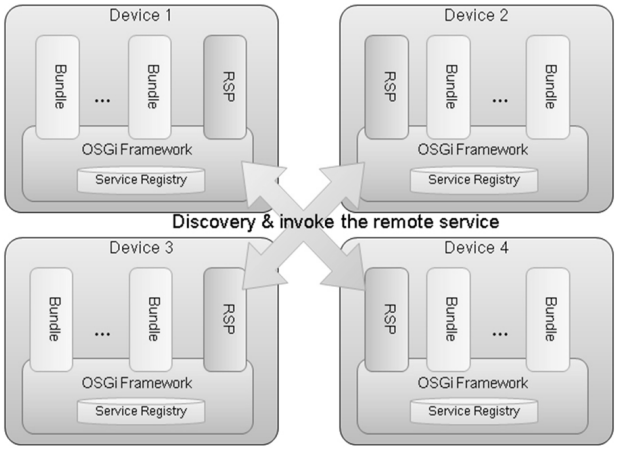
번들 내부의 클래스를 패키지 단위로 모듈화하여 공유(export/import)하고, 번들의 동적인 설치(installed), 구동(started), 갱신(updated), 정지(stopped), 삭제(uninstalled)와 같은 라이프 사이클 관리를 통해 번들간의 의존성을 제공한다. 또한 번들은 어느 때나 설치 및 삭제가 가능하며, OSGi 프레임워크 안에 있는 다른 번들에게 서비스를 제공할 수 있다. 서비스 레지스트리는 이러한 번들간의 동적인 서비스 관계를 관리하며, 서비스 레지스트리에 서비스를 검색하거나 등록/삭제하고자 할 때 관련 번들에게 알려주는 기능을 한다. 번들과 프레임워크간의 상호 작동은 (그림 1)과 같다.



(그림 1) 번들과 프레임워크간의 상호 작동 관계

4. RSP(Remote Service Provider) 설계 및 구현

한 대의 자바 가상 머쉰에서 동작하는 OSGi 프레임워크에서 번들은 (그림 1)과 같이 로컬 OSGi 서비스 레지스트리에 등록된 서비스만을 공유할 수 있다. 반면, 본 논문에서 제안하는 RSP는 (그림 2)와 같이 OSGi가 탑재된 장치들이 네트워크로 연결되어 있는 분산 컴퓨팅 환경에서 원격 OSGi 프레임워크가 구동하는 서비스를 OSGi 프레임워크들이 서로 공유할 수 있게 한다. RSP는 OSGi 번들이며, 로컬



(그림 2) RSP에 의한 분산된 OSGi 프레임워크 구조

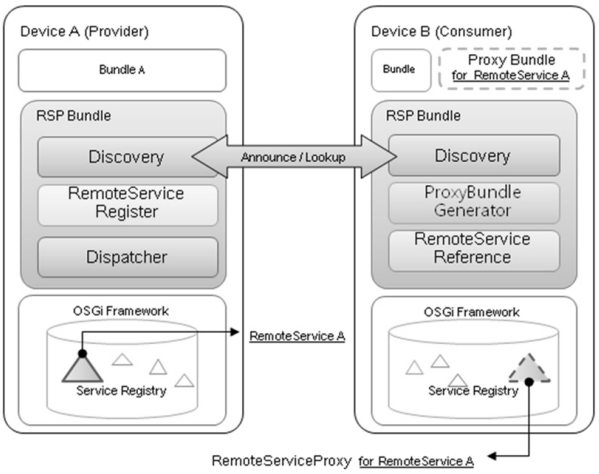
OSGi 번들은 RSP를 사용하여 원격 OSGi 서비스를 쉽게 호출하여 사용할 수 있게 된다. 따라서 본 논문에서는 투명성과 신뢰성, 그리고 일관성의 특성을 고려하여 RSP를 설계한다.

- 투명성: OSGi 프레임워크는 원격 OSGi 서비스를 로컬 OSGi 서비스와 동일한 방법으로 호출한다. 따라서 RSP를 사용하여 원격 서비스를 사용하고자 하는 번들 개발자는 로컬 서비스 호출과 동일한 구문(syntax)을 사용한다. 또한 RSP는 위치 투명성을 제공한다. 즉, RSP 클라이언트는 호출하는 서비스가 로컬 서비스인지 또는 원격 서비스인지에 대한 위치 정보에 상관없이 서비스를 호출할 수 있으며, 서버측 원격 서비스 또한 서비스 요청자의 위치 정보에 상관없이 동일하게 서비스를 제공한다. 따라서 RSP는 클라이언트, 서버측 모두에게 투명성을 제공하기 위한 서비스 등록 및 발견, 메시지 처리 방법을 제공한다.

- 신뢰성: OSGi 프레임워크에서는 동적인 번들로 인해서 로컬 서비스 레지스트리에 서비스의 등록, 등록 해제, 갱신이 빈번히 발생하게 된다. 이러한 동적인 OSGi 서비스 상태가 로컬 OSGi 서비스 레지스트리에 정확히 반영되므로 OSGi 프레임워크는 신뢰성을 제공한다. 따라서 분산된 OSGi 프레임워크에서도 원격 OSGi 프레임워크에서 발생하는 원격 OSGi 서비스의 등록, 등록 해제, 갱신과 같은 상태 변화를 로컬 OSGi 프레임워크가 즉각 통보 받을 수 있도록 지원한다. 이렇게 원격 서비스에 대한 신뢰성을 보장함으로써, 로컬 OSGi 프레임워크가 원격 서비스의 등록이 해제되었는데도 불구하고 원격 서비스를 호출하는 상황이 발생하지 않게 한다.

- 일관성: RSP는 원격 OSGi 서비스를 지원하기 위해 OSGi 자체 기술만을 사용하며 표준 OSGi 프레임워크의 스펙을 그대로 따른다. 원격 OSGi 서비스를 지원하기 위해 표준 스펙을 변경하면 이미 구현된 OSGi 서비스를 사용하기 위해 다시 구현하여야 한다. 따라서 RSP는 원격 서비스의 등록 및 검색, 바인딩, 호출 등의 모든 기능이 표준 OSGi 스펙을 그대로 따르므로 기존에 사용하던 OSGi 서비스를 그대로 재사용할 수 있다. RSP는 OSGi 번들, 서비스 레지스트리, 서비스 트랙커(ServiceTracker), 번들 라이프 사이클 관리와 같은 OSGi 자체 기술만을 사용한다. 따라서 기존의 분산 미들웨어를 사용하는 기술들보다 컴퓨팅 자원 소모가 적고 원격 OSGi 서비스 정보 표현도 OSGi 프레임워크의 서비스 정보 표현 방식인 키/값(key/value) 형식과 그대로 일치하므로 서비스 변환 작업이 불필요하다.

원격 OSGi 서비스의 투명성 및 신뢰성, OSGi 기술과의 일관성을 제공하도록 설계된 RSP의 구조는 (그림 3)과 같다. RSP는 OSGi 번들로서 RemoteServiceRegister, RemoteService-Reference, Discovery, ProxyBundleGenerator, Dispatcher 서비스로 구성된다. 4.1절부터 4.5절에서는 RSP 번들의 각 서비스들의 설계와 구현에 대하여 자세히 논한다.



(그림 3) RSP(Remote Service Provider) 구조

4.1 RemoteServiceRegister

RSP의 RemoteServiceRegister 서비스는 원격으로 공유할 서비스를 로컬 OSGi 서비스 레지스트리에 등록하는 서비스이다. 등록된 원격 서비스는 RSP의 Discovery 서비스를 통해서 원격 서비스를 광고하거나 룩업(lookup) 요청에 응답할 수 있게 된다. RSP는 원격으로 공유할 서비스를 정의하기 위해서 OSGi의 서비스 속성을 사용한다. 즉, 원격 서비스를 나타내는 속성으로 'service.remote=ture' 라는 새로운 원격 속성을 정의한다. RemoteServiceRegister 서비스는 이렇게 정의된 원격 속성을 가진 원격 서비스를 로컬 OSGi 서비스 레지스트리에 등록한다.

OSGi 서비스의 표준 서비스인 EventAdmin 서비스는 번들간의 상호작용을 제공하기 위하여 이벤트를 생성하고, 이벤트를 처리하는 EventHandler를 정의하여 등록하는 서비스이다. RemoteServiceRegister 서비스는 EventAdmin 서비스를 사용하여 등록된 원격 서비스에 대한 룩업 이벤트가 발

생했을 때, 룩업 요청 이벤트를 처리할 수 있는 이벤트 핸들러를 생성하여 OSGi 프레임워크에 등록한다.

한편 OSGi 서비스 중 ServiceTracker는 로컬 OSGi 서비스 레지스트리를 지속적으로 모니터링하여 서비스의 등록, 등록 해제, 갱신과 같은 상태 변화가 일어났을 경우 정의된 행동을 실행하는 서비스이다. RSP의 Discovery 서비스는 ServiceTracker를 상속받은 RemoteServiceTracker을 사용하여 원격 서비스 등록을 확인했을 때, RSP의 Discovery 서비스에게 원격 서비스가 등록되었음을 알려서 새로 등록된 원격서비스를 원격 OSGi 프레임워크들에게 광고하게 한다.

4.2 RemoteServiceReference

OSGi 프레임워크의 번들이 원격에 있는 OSGi 서비스를 사용하고자 할 때, 먼저 RSP의 RemoteServiceReference 서비스를 사용하여 원격 서비스에 대한 ServiceReference를 획득해야 한다. RemoteServiceReference 서비스를 통해 리턴되는 ServiceReference는 OSGi의 ServiceReference와 동일한 행태로서 등록된 서비스에 대한 아이디, 서비스 오브젝트와 같은 정보를 포함하며, 추가적으로 원격 서비스에 대한 IP 주소, 포트번호를 포함한다. RemoteServiceReference 서비스는 EventAdmin 서비스를 이용함으로써 찾고 있는 원격 서비스에 대한 발견을 알리는 광고 이벤트를 처리할 수 있는 EventHandler를 생성하여 OSGi 프레임워크에 등록한다. RemoteServiceReference 서비스에 의해 등록된 EventHandler는 원격 서비스에 대한 정보를 받아 ProxyBundle를 생성하는 ProxyBundleGenerator을 생성한다.

RemoteServiceReference 서비스는 원격 서비스에 대한 ServiceReference를 획득하기 위하여 원격 서비스에 대한 프록시 서비스가 등록되어 있는지를 검사한다. 이용하고자 하는 원격 서비스에 대한 서비스가 등록되어 있다면, 바로 원격 서비스를 사용하고자 하는 번들은 서비스에 대한 ServiceReference를 획득한다. 만약 이용하고자 하는 원격 서비스에 대한 서비스가 등록되어 있지 않다면, Discovery 서비스에 원격 서비스를 룩업 요청한 후에 RemoteServiceReference는 로컬 OSGi 서비스 레지스트리에 요청한 원격 서비스의 프록시 서비스가 등록되기를 기다리게 된다. Discovery 서비스가 요청한 원격 서비스를 발견하면, 발견한 원격 서비스 정보를 ProxyBundleGenerator에게 알린다. ProxyBundleGenerator는 ProxyBundle을 생성하여 구동시키고, 구동된 ProxyBundle은 원격 서비스에 대한 프록시(Proxy)를 생성하여 로컬 OSGi 서비스 레지스트리에 등록한다. 해당 프록시 서비스가 등록되면 RemoteServiceReference에 통지되고, RemoteServiceReference는 원격 서비스를 사용하고자 하는 번들에게 등록된 프록시 서비스의 ServiceReference를 알려준다.

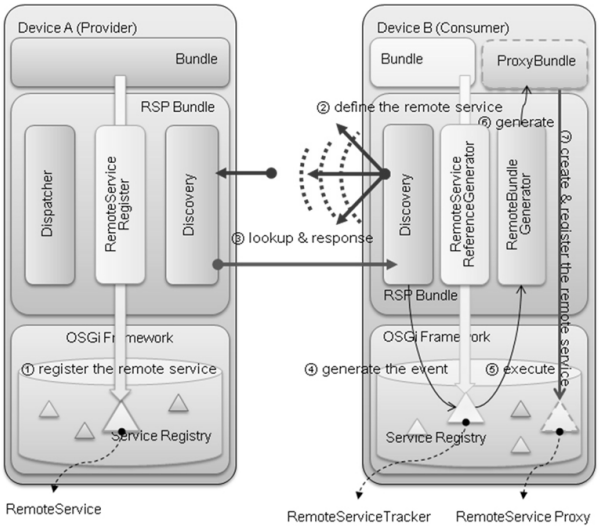
4.3 Discovery

RSP의 Discovery 서비스는 원격 OSGi 서비스를 광고하거나 필요한 원격 OSGi 서비스를 찾기 위하여 사용한다. RSP의 Discovery 서비스는 P2P 구조를 갖는다. 즉, RSP를 실행하는 각각의 OSGi 프레임워크는 피어(peer)가 되고, 각 피어들은 원격 서비스 레지스트리로 로컬 OSGi 서비스 레지스트리를 이용하는 구조이다. RSP의 Discovery 서비스는 ServiceTracker 서비스, 멀티캐스트 통신 채널, 유니캐스트 통신 채널, EventAdmin 서비스로 구성된다. RSP의 Discovery 서비스는 로컬 OSGi 서비스 레지스트리에서 원격 OSGi 서비스의 등록, 삭제, 갱신을 실시간으로 끊임없이 모니터링하기 위하여 OSGi의 ServiceTracker를 상속받아 구현한 RemoteServiceTracker를 사용한다. 또한 RSP는 P2P를 지원하기 위한 멀티캐스트 통신 채널을 가지고 있고, Discovery 서비스의 룩업 응답 메시지를 전송할 때 메시지 전송의 효율성을 위하여 유니캐스트를 사용하므로 TCP 통신 모듈을 가진다. Discovery 서비스는 멀티캐스트 채널을 통해 새로운 원격 서비스가 등록되었다는 것이 알려졌을 때, 새로운 원격 서비스의 발견을 알리는 이벤트를 발생시키기 위해 EventAdmin 서비스를 사용한다.

원격으로 OSGi 서비스를 제공하고자 하는 번들이 RSP의 RemoteServiceRegister 서비스를 사용하여 로컬 OSGi 서비스 레지스트리에 원격 서비스를 등록하면, Discovery 서비스의 RemoteServiceTracker는 원격 서비스의 등록을 발견하게 된다. RSP의 Discovery 서비스는 새로 등록된 원격 서비스를 다른 피어들에게 광고하기 위하여 서비스 이름과 서비스 속성, 즉 OSGi 프레임워크가 서비스를 등록할 때 부여한 서비스 아이디, 서비스 인터페이스, IP 주소, 포트 번호를 메시지에 실어 멀티캐스트로 모든 피어에 전송한다. 이때, IP 주소는 원격 서비스를 제공하는 OSGi 프레임워크 머쉰의 IP 주소이고, 포트 번호는 Dispatcher가 원격 서비스 요청을 받기 위해 사용하는 포트 번호이다. 멀티캐스트로 새로 등록된 원격 서비스의 광고 메시지를 받은 피어의 Discovery 서비스는 원격 서비스 정보를 포함하는 광고 이벤트를 EventAdmin 서비스를 이용하여 발생시킨다. 발생된 광고 이벤트를 처리하는 핸들러가 RemoteServiceReference에 의해서 등록되어 있다면, 이 핸들러는 원격 서비스 정보를 이용하여 ProxyBundleGenerator을 구동시킨다.

원격 OSGi 서비스를 제공하는 서버측 Discovery 서비스의 RemoteServiceTracker는 원격 서비스의 등록뿐만 아니라, 원격 서비스의 삭제, 갱신 시에도 다른 피어들에게 원격 서비스의 상태변화 정보를 멀티캐스트한다. 한편 마찬가지로 원격 OSGi 서비스의 클라이언트측에 해당하는 Discovery 서비스는 원격 서비스의 삭제, 갱신 메시지를 받아 이벤트를 발생하여 해당 원격 서비스에 대응하는 ProxyBundle의 라이프 사이클을 관리한다.

한편 (그림 4)와 같이 원격 OSGi 프레임워크가 원격 서비스를 제공가능 할 때, 로컬 OSGi 프레임워크에서 원격 서비스를 사용하고자 하는 번들이 구동되어 Discovery 서비스에게 원격 서비스를 요청(lookup)하는 경우도 발생한다. 원격 OSGi 서비스를 사용하고자 하는 번들이 RemoteServiceReference 서비스에게 해당 서비스를 요청하면, RemoteServiceReference는 우선 로컬 OSGi 서비스 레지스트리에서 해당 원격 서비



(그림 4) 원격 서비스를 찾기 위한 룩업(lookup) 과정

스의 프록시 서비스를 찾고, 프록시 서비스를 발견하지 못했을 경우 Discovery 서비스에 룩업을 요청한다. Discovery 서비스는 요청된 원격 서비스에 대한 룩업 메시지를 만들어 다른 피어들에게 멀티캐스트한다. 이 때 룩업 요청 메시지에는 Discovery 서비스의 유니캐스트 채널 정보가 포함되며, 이 TCP 통신 정보는 원격 서비스를 가지고 있는 피어가 룩업 응답 메시지를 보낼 때 사용한다.

원격 서비스 룩업 요청 메시지를 받은 피어의 Discovery 서비스는 요청 서비스 정보를 포함하는 룩업 이벤트를 발생한다. 발생한 룩업 이벤트를 처리하는 RemoteServiceRegister가 있다면, RemoteServiceRegister에 등록되어 있는 요청 서비스 정보인 서비스 이름과 서비스 속성를 포함한 룩업 응답 메시지를 TCP 통신을 사용하여 응답한다. 룩업 응답 메시지의 목적지는 룩업 요청 메시지에 포함되어 있는 TCP 통신 정보를 사용하여 룩업 서비스 요청자에게 전송한다.

4.4 ProxyBundleGenerator

RSP의 ProxyBundleGenerator는 원격 OSGi 서비스에 대한 프록시 서비스와 이 프록시 서비스를 포함하는 ProxyBundle을 동적으로 생성시키는 서비스이다. OSGi에서 서비스는 번들이라는 소프트웨어 단위에 포함되고, 번들의 설치, 구동, 삭제, 갱신과 같은 번들 라이프 사이클은 OSGi 프레임워크가 관리한다. ProxyBundleGenerator가 생성한 ProxyBundle의 라이프 사이클 또한 로컬 OSGi 프레임워크에서 관리한다. 따라서 RSP는 로컬 Discovery 서비스의 RemoteServiceTracker를 통해 원격 OSGi 프레임워크에서 발생하는 서비스 상태 변화에 대한 정보를 받아서 ProxyBundle의 라이프 사이클을 관리하므로 원격 서비스에 대한 신뢰성을 보장할 수 있다.

ProxyBundleGenerator가 생성한 ProxyBundle은 원격 서비스 오브젝트와 1:1 관계를 가진다. 또한 생성된 ProxyBundle은 서비스 오브젝트로서 원격 서비스에 대한 프록시 서비스인 RemoteServiceProxy, 번들을 구동하기 위한 Activator 클래스, 번들 정보를 포함하고 있는 MANIFEST 파일을 가진다. ProxyBundleGenerator는 Discovery 서비스가 발생시킨 새로운 원격 서비스에 대한 광고 이벤트로부터 원격 서비스에 대한 서비스 오브젝트에 대한 인터페이스 이름, 서비스 아이디, IP 주소, 포트 번호를 받고, 이를 기반으로 RemoteServiceProxy를 생성한다. ProxyBundle이 OSGi 프레임워크에서 구동될 때, Activator 클래스는 RemoteServiceProxy를 로컬 OSGi 서비스 레지스트리에 실제 서비스와 같이 등록한다.

로컬 OSGi 서비스 레지스트리에 등록된 RemoteServiceProxy를 통해서 원격 OSGi 서비스는 그 위치와 관계없이 로컬 OSGi 서비스처럼 호출되어 사용된다. 따라서 원격 OSGi 서비스에 대한 프록시 서비스가 호출되면 네트워크를 통해 실제 서비스를 가지고 있는 원격 OSGi 프레임워크의 Dispatcher에 호출 메시지가 전달된다. RemoteServiceProxy는 내부적으로 원격 OSGi 프레임워크에서 동작중인 Dispatcher와 TCP통신을 한다. Dispatcher는 실제 OSGi 서비스를 호출하여 실행하고, 그 실행 결과를 다시 네트워크를 통해 RemoteServiceProxy에게 반환한다.

본 논문에서는 ProxyBundleGenerator가 원격 OSGi 서비스에 해당하는 ProxyBundle의 코드를 동적으로 생성하도록 ASM [15] 라이브러리를 사용하였다. ASM은 자바 클래스들을 동적으로 생성하고 조작할 수 있도록 라이브러리를 제공하는 도구이다.

4.5 Dispatcher

RSP의 Dispatcher는 원격 서비스를 제공하는 서버측 OSGi 프레임워크에서 클라이언트의 원격 서비스 요청을 받아들이고 처리하는 데몬 서비스이다. RSP 번들이 구동되면 Dispatcher 서비스가 실행되어 원격 서비스에 대한 클라이언트 호출 요청을 기다리게 된다.

클라이언트측 RemoteServiceProxy는 원격 서비스에 대한 서비스 아이디, 서비스 인터페이스 이름, 아큐먼트를 메시지에 포함하여 TCP 통신을 통해 서버측 Dispatcher에게 보낸다. Dispatcher는 클라이언트의 원격 서비스 요청 메시지를 해석하여 원격 서비스로 정의된 해당 로컬 OSGi 서비스를 호출한다. 서비스 호출 후 결과값은 호출 요청을 한 RemoteServiceProxy에게 보내어 서비스 요청에 대한 응답을 완료한다.

5. 성능 평가

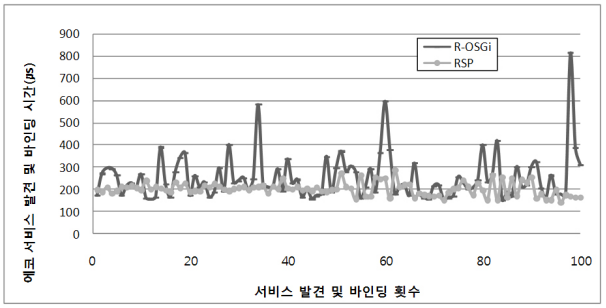
5장에서는 4장에서 설계한 RSP의 성능을 평가하고, 2.3절 관련 연구에서 언급한 R-OSGi의 성능과 비교 분석하였다. RSP와 마찬가지로 R-OSGi는 분산 OSGi 프레임워크들의 서비스를 서로 공유하기 위하여 OSGi 자체 기술만을 사용한다. 따라서 관련연구에서 언급한 Jini, UPnP, JXTA 또는 Web Services 같은 기존의 분산 미들웨어를 이용하는 기술들과 비교했을 때, RSP와 R-OSGi는 서비스의 변환과정이

필요없어 빠르며 컴퓨팅 자원의 소모가 적은 기술이다.

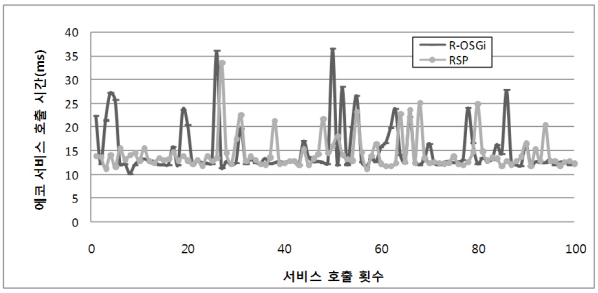
성능 평가를 위하여 Intel Core2 (1.86GHz, 1.87GHz) CPU, 2G RAM이 장착된 PC 1대를 서비스 사용자 머쉰으로, Intel Core2 Quad (2.33GHz, 2.33GHz) CPU, 3G RAM이 장착된 PC 1대를 서버 제공자 머쉰으로 각각 구성하였다. 각각의 PC에는 Windows VISTA, JDK 1.6과 OSGi 프레임워크로 이클립스의 기반이 되는 Equinox를 설치하였다.

성능 평가에 사용한 원격 OSGi 서비스는 EchoService이다. EchoService는 클라이언트가 스트링 아큐먼트에 값을 설정해 원격 EchoService를 호출하면, 다시 동일한 스트링을 클라이언트에 돌려보내는 서비스이다. 먼저 서비스 제공자 PC에 EchoService을 원격으로 서비스할 수 있도록 RSP를 이용하여 Provider 번들을 구현하고 OSGi 프레임워크에서 구동하였다. 그리고 클라이언트로 RSP을 이용한 Request 번들을 만들어 서비스 사용자 PC에서 실행되는 OSGi 프레임워크에서 구동시킨 후, EchoService를 사용하기 위한 서비스 발견 및 바인딩 시간 그리고 EchoService의 서비스 호출(Invocation) 시간을 각각 측정하였다. 서비스 발견 및 바인딩 시간은 서비스를 발견과 서비스 인터페이스를 받아 서비스 호출에 필요한 프록시를 생성하여 로컬 OSGi 프레임워크에 등록하는데 걸리는 시간이다. RSP와 성능 비교를 위하여 R-OSGi도 RSP와 동일한 실험 환경에서 EchoService를 구현하고 구동하여 서비스 발견 및 바인딩 시간 그리고 호출 시간을 측정하였다. EchoService의 아큐먼트의 크기가 Null, 512바이트, 1024바이트 일 때 각각 그 성능을 측정하였다.

(그림 5)는 EchoService의 아규먼트가 null일 때, RSP와 R-OSGi의 디스커버리 시간 및 바인딩 시간을 100회 측정하여 도식화한 것이다. RSP의 평균 서비스 발견 및 바인딩 시간은 평균 약 199.7ms이고, R-OSGi의 평균 시간은 약 249.1ms로 측정되었다. RSP의 서비스 발견 및 바인딩 시간은 R-OSGi보다 약 0.5배 빠른 성능 결과를 얻었다. 이것은 R-OSGi가 중앙 집중적인 서비스 레지스트리를 사용하여 서비스를 발견 후 원격 서비스를 포함하는 OSGi 프레임워크에 연결되는 것과는 달리, RSP는 서비스 발견을 위하여 P2P 방식의 매커니즘을 사용하므로 서비스 발견 즉시 원격 서비스를 가진 피어와 연결되기 때문에 RSP의 바인딩 시간이 R-OSGi 바인딩시간보다 빠르게 측정된 것이다.



(그림 5) RSP와 R-OSGi의 서비스 발견 및 바인딩 시간 비교



(그림 6) RSP와 R-OSGi의 서비스 호출 시간 비교

(그림 6)은 EchoService의 아규먼트가 null일 때, RSP와 R-OSGi의 호출 시간을 100회에 걸쳐 측정한 실험결과를 나타낸다. RSP의 평균 EchoService 호출 시간은 평균 약 14.4ms이고, R-OSGi는 평균 약 14.9ms로 측정되었다. EchoService 호출 시간은 RSP와 R-OSGi 모두가 프록시를 이용하여 원격 서비스를 호출하기 때문에 거의 비슷한 결과를 얻었다. 100회의 실험결과를 통해 RSP와 R-OSGi가 원격 아규먼트가 null인 EchoService를 호출하여 실행 결과를 돌려받는데 걸린 총 소요 평균 시간은 <표 1>과 같다. RSP는 평균 약 214.1ms, R-OSGi는 약 263.9ms의 시간이 걸렸으므로 RSP가 R-OSGi보다 약 0.5배 빠른 성능을 보인다. <표 1>에는 EchoService의 아규먼트의 512byte일 때, 1024byte일 때의 각각의 성능을 나타내고 있다.

〈표 1〉 원격 서비스를 이용하기 위한 평균 시간 (ms)

구 분	서비스 검색 및 바인딩		서비스 호출		총 소요 시간	
	R-OSGi	RSP	R-OSGi	RSP	R-OSGi	RSP
NULL	249.1	199.7	14.9	14.4	263.9	214.1
512byte	252.7	217.3	18.8	18.2	271.5	235.4
1024byte	298.4	226.2	22.1	20.8	320.4	247.0

6. 결론 및 향후 과제

OSGi 프레임워크가 유비쿼터스 컴퓨팅 환경을 구성하는 이질적이고 다양한 성능의 장치들에 탑재되면서, OSGi 프레임워크들 사이의 서비스 공유가 필요하게 되었다. 분산된 OSGi 프레임워크들 간의 서비스 공유를 위하여 기존의 분산 미들웨어 기술인 P2P 기반의 JXTA 및 Web Services 기술을 활용하는 연구들이 있었으나, 서비스 표현방식이 서로 다르기 때문에 서비스를 변환하는 과정이 부가적으로 필요하고, OSGi 프레임워크에 기존 분산기술을 번들로 구현하여 탑재하므로 컴퓨팅 자원 소모가 많다는 문제점이 있다. 이를 해결하기 위하여 R-OSGi 같이 OSGi 자체 기술만을 사용하는 OSGi 미들웨어가 개발되었지만, R-OSGi는 중앙 집중적인 서비스 레지스트리를 사용함으로써 병목현상이나 SPOF (Single Point of Failure)이 발생할 수 있는 문제점을

가질 수 있다.

본 논문에서는 분산된 OSGi 프레임워크들간의 서비스 공유를 위하여 OSGi 기술을 그대로 활용하는 RSP OSGi 번들을 제안하였다. RSP는 P2P방식의 서비스 발견 메커니즘을 사용하여 중앙집중적인 서비스 레지스트리 방식의 문제점을 해결하였다. 또한 원격 OSGi 서비스에 대한 투명성을 제공하고, 원격 OSGi 서비스의 상태 변화를 즉각 통보하여 원격 서비스의 신뢰성을 보장한다. OSGi 자체 기술만으로 설계되었으므로 서비스 변환과 같은 부가적인 작업이 불필요하며, JXTA나 Web Services 같은 기존 분산 미들웨어를 활용하는 기술들과 비교했을 때 컴퓨팅 자원소모가 적다는 특성이 있다.

현재 OSGi 기술은 통신, 자동차, 공장, 사무실, 가정 등 그 적용 도메인이 점차 다양해지며 보편화되는 추세이다. 따라서 원격 OSGi 프레임워크의 서비스 공유 시 보안뿐만 아니라 서로 다른 도메인간의 상호운용성 또한 더 고려되야 할 필요성이 있다. 또한 RSP는 유비쿼터스 홈 도메인에서 OSGi가 탑재된 기기들간의 상호운영을 제공하기 위해 연구되었으므로 가정에서의 일상 생활에 적용시킬 수 있는 유비쿼터스 어플리케이션을 개발할 예정이다.

참 고 문 헌

[1] About the OSGi Alliance, http://www.osgi.org/About/HomePage.

[2] Choonhwa Lee, Nordstedt. D, Helal. S, “Enabling smart spaces with OSGi,” Pervasive Computing, IEEE, Volume 2, Issue 3, pp. 89-94, July-Sept, 2003.

[3] Haitao. Zhang, Fei-Yue. Wang, Yunfeng. Ai, “An OSGi and agent based control system architecture for smart home,” Networking, Sensing and Control, 2005. Proceedings, pp.13-18, 2005.

[4] Gu. T, Pung. H.K, Zhang. D.Q, “Toward an OSGi-based infrastructure for context-aware applications,” Pervasive Computing, IEEE, Vol. 3, Issue 4, pp.66-74, Oct.-Dec., 2004.

[5] Hiroo Ishikawa, Yuuki Ogata, Kazuto Adachi, Tatsuo Nakajima, “Building Smart Appliance Integration Middleware on the OSGi Framework,” pp.139-146, ISORC ’04.

[6] Deepak Kamlani, “OSGi Alliance to Participate in CES Track Session,” OSGi Alliance, Jan., 2004.

[7] OSGi Alliance, “About the OSGi Service Platform,” June, 2007.

[8] OSGi Service Platform Core Specification. Release 4, Version 4.1, April, 2007.

[9] Why did we let Jini out of OSGi R4, http://www.osgi.org/blog/2005/11/why-did-we-let-jini-out-of-osgi-r4.html.

[10] JXTA - Project JXTA, 2003. http://www.jxta.org.

[11] A Ferscha, M Hechinger, R Mayrhofer, R Oberhauser, “A Light-Weight Component Model for Peer-to-Peer Applications,” Distributed Computing Systems Workshops, 2004.

[12] CL Wu, CF Liao, LC Fu, “Service-Oriented Smart-Home Architecture Based on OSGi and Mobile-Agent Technology,” IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Vol.37, No.2, pp.193-205, March, 2007.

[13] Erik guttman, “Service Location Protocol : Automatic Discovert of IP Network Services,” IEEE Internet Computing Magazine, pp.71-80, July-August, 1999.

[14] SLP White Paper, http://playground.sun.com/srvloc/slp_white_paper.html.

[15] J. S. Rellermeyer, G. Alonso, “Service Everywhere: OSGi in Distributed Environments,” In EclipseCon, 2007.

[16] ASM, http://asm.objectweb.org/index.html.

윤 기 현

e-mail : khyun@ss.ssu.ac.kr

2007년 목원대학교 컴퓨터공학과(학사)

2007년~현 재 숭실대학교 컴퓨터학과 석사과정

관심분야 : 유비쿼터스 컴퓨팅, RFID, USN

김 은 회

e-mail : ehkim@ss.ssu.ac.kr

1989년 숭실대학교 전자계산학과(학사)

1996년 숭실대학교 컴퓨터학과(공학석사)

2006년 숭실대학교 컴퓨터학과(공학박사)

2008년~현 재 숭실대학교 정보미디어 기술연구소 전임연구원

관심분야 : 병렬/분산처리, 유비쿼터스 컴퓨팅, RFID

최 재 영

e-mail : choi@ssu.ac.kr

1984년 서울대학교 제어계측공학과(학사)

1986년 미국 남가주대학교 컴퓨터공학(석사)

1991년 미국 코넬대학교 컴퓨터공학(박사)

1992년~1994년 미국 국립 오크리지연구소 연구원

1994년~1995년 미국 테네시 주립대학교 연구교수

2001년~2002년 미국 국립 수퍼컴퓨팅 응용센터 (NCSA) 초빙연구원

1995년~현 재 숭실대학교 컴퓨터학부 교수

관심분야 : 병렬/분산처리, 고성능컴퓨팅(HPC), 유비쿼터스컴퓨팅