원격 OSGi 서비스의 상호운용 기술 설계 및 구현☆

Design and Implementation of the Interoperability method for the Remote OSGi services

김 은 회* 윤 기 현** 최 재 영***
Eunhoe Kim Kihyun Yun Jaeyoung Choi

요 약

유비쿼터스 환경의 OSGi는 디지털 이동 전화, 차량, 텔레메틱스, 임베디드 가전, 가정용 게이트웨이, 산업용 컴퓨터, 데스크탑 컴퓨터, 고성능 서버에 이르기까지 그 적용범위가 확대되고 있다. 따라서 다양한 장비에 탑재된 OSGi 프레임워크의 서비스들을 상호운용할 수 있는 기술이 필요하게 되었다. 본 논문에서는 분산 OSGi 프레임워크에서 원격 서비스의 상호운용을 지원하기 위하여 대표적인 분산 미들웨어 기술인 RMI 패러다임을 적용한 원격 OSGi 서비스 상호운영 방안을 제안한다. 제안하는 원격 OSGi 서비스 상호운용 방안은 OSGi 표준 기술을 활용 및 확장하여 서비스 지향적인 OSGi 아키텍처에 부합하는 원격서비스의 등록 및 발견, 접근 방법을 제공하며, 동적으로 변하는 원격 서비스들의 속성을 일관성 있게 유지하여 원격서비스에 대한 신뢰성을 지원한다. 또한 동적으로 프락시 번들 및 프락시 서비스를 생성함으로써 원격 OSGi 서비스의 위치 투명성을 지원하는 특징을 가진다.

ABSTRACT

In ubiquitous computing environment, OSGi has applied to many areas such as digital mobile phones, vehicles, telematics, embedded appliances, residential gateways, industrial computers, desktop PCs, and high-end servers including mainframes. Therefore, interoperability is required for remote OSGi services which are built on various devices. In this paper, we proposed a method which was able to interoperate remote OSGi services using RMI paradigm. RMI is a representative middleware technology in distributed computing environment. The suggested method is based on the standard OSGi technology. It is possible to provide remote OSGi service registration, finding, and binding methods which were suitable for the OSGi service-oriented architecture. We also provided reliability of the dynamic remote OSGi services by maintaining consistent properties of them, and we could provide location transparency of the remote OSGi services by generating proxy bundles and proxy services dynamically.

☞ keyword : OSGi, interoperability(상호운용성), RMI, proxy(프록시), remote service(원격서비스)

1. 서 론

네트워크로 컴퓨팅 자원들을 공유하여 작업을 수행하는 분산 컴퓨팅 환경은 1968년도 인터넷 프로토콜이 개발된 이후로 급격히 발전하여 오늘날에 이르렀다. 현재 유무선 네트워크로 데스크탑 PC, 이동 장비, 정보가전기기, 센서 등을 연결하는 유비쿼터스 컴퓨팅 환경은 사용자에게 언제 어디서든, 제 때에 가장 최적의 서비스를 제공하고자 한다. 따라서 프로세스간 통신뿐만 아니라, 복잡한 유비쿼터스 환경의 다양한 장비에서 수행하는 서비스들까지 쉽게 상호운용할 수 있는 미들웨어 기술이 필요하게 되었다.

현재 RMI(Remote Method Invocation)는 대표적

* 정 회 원 : 숭실대학교 지능형로봇연구소 연구원
ehkim@ss.ssu.ac.kr
** 정 회 원 : 숭실대학교 연구원
yunkiv@gmail.com
*** 종신회원 : 숭실대학교 IT대학 컴퓨터학부 교수
choi@ssu.ac.kr(교신저자)
[2010/07/30 투고 - 2010/08/12 심사(2010/12/01 2차) - 2010/12/22 심사완료]
☆ 본 논문은 한국연구재단의 2009년 기초연구사업(2009-0074852)의 지원을 받아 수행되었습니다.

인 분산 미들웨어 기술이다 [1]. 프로세스간 정보를 교환할 때 발생하는 마샬링 및 언마샬링, 메시지교환 프로토콜, 운영체제의 상이성 등을 프로그램 개발자가 신경쓰지 않도록 처리해주고, 원격 오브젝트의 메소드를 로컬 오브젝트의 메소드를 호출하듯 사용할 수 있도록 해준다. RMI 미들웨어를 사용함으로써 어플리케이션 개발자는 프로그램을 쉽고 빨리 개발할 수 있기 때문에, 현재 RMI는 분산 시스템 개발에 많이 사용하는 대표적인 미들웨어 기술로 자리매김하게 되었다.

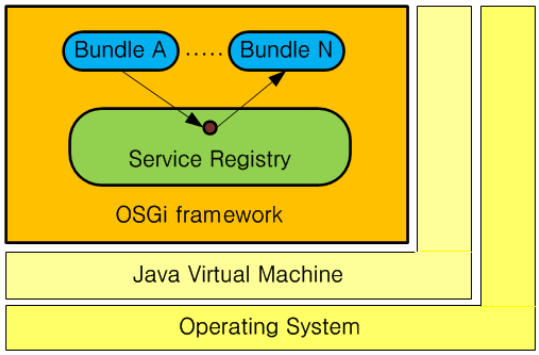
한편, 2000년에 KT, 노키아, 모토롤라, 삼성전자, 히타쯔, IBM, 오라클, HP 등과 같은 유수한 통신, 가전, IT 회사들이 참여하는 OSGi Alliance에서 일종의 미들웨어 프레임워크로서 OSGi (Open Service Gateway Initiative) [2]를 발표하였다. 초기 OSGi는 홈 네트워크의 표준으로 인식되었고 큰 주목을 받는 기술이 아니었지만, 점차 미들웨어 기능이 보완됨에 따라 현재는 유비쿼터스 환경의 자바 가상 머쉰이 동작하는 다양한 성능의 디바이스에서 서비스의 공유와 배포를 위한 미들웨어로 크게 각광받는 기술이 되었다. 현재 OSGi는 홈 네트워크 뿐만 아니라, 디지털 이동 전화, 차량, 텔레메틱스, 임베디드 가전, 가정용 게이트웨이, 산업용 컴퓨터, 데스크탑 컴퓨터, 고성능 서버에 이르기까지 그 적용범위가 확대되고 있다 [2, 3]. 그러므로 다양한 장비에 탑재된 OSGi 프레임워크의 서비스들을 상호운용 할 수 있는 기술의 필요성이 점차 증대되고 있는 상황이다 [4, 5].

따라서 본 논문에서는 분산 OSGi 프레임워크에서 원격 서비스의 상호운용이 가능하도록 대표적인 분산 미들웨어 기술인 RMI 패러다임을 적용하여 원격 OSGi 서비스 상호운영 방안을 제안한다. 제안하는 원격 OSGi 서비스 상호운용 방안은 OSGi 표준 기술을 활용 및 확장하여 서비스 지향적인 OSGi 아키텍처에 부합하는 원격서비스의 등록 및 발견, 접근 방법을 제공하며, 동적으로 변하는 원격 서비스들의 속성을 일관성 있게 유지하여 원격서비스에 대한 신뢰성을 지원한다. 또한 동적으로 프락시 번들 및 프락시 서비스를 생성함으로써 원격 OSGi 서비스의 위치 투명성을 지원하는 특징을 가진다.

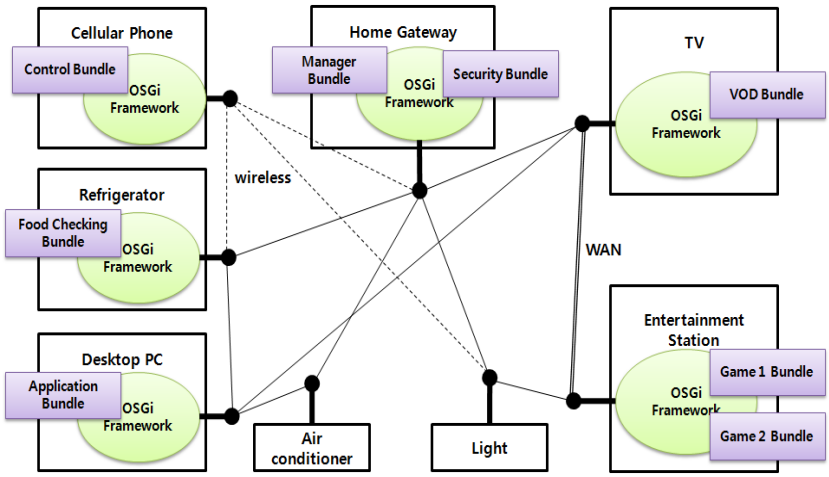
본 논문의 구성은 다음과 같다. 2장에서는 분산 OSGi 프레임워크의 특징과 RMI 패러다임의 적용 방안을 논한다. 3장에서는 원격 OSGi 서비스 상호운영 시스템 설계와 구현을 설명하고, 4장에서는 구현한 시스템의 성능을 평가한다. 5장에서는 원격 OSGi 서비스 상호운용 시스템의 응용을 제시하며, 6장에서 관련 연구를 논하고, 7장에서는 결론과 향후연구 과제를 논한다.

2. 분산 OSGi 프레임워크에서 RMI 패러다임 적용 방안

OSGi는 자바를 위한 동적모듈 관리시스템이다. 핵심 기술인 OSGi 프레임워크는 OSGi에서 어플리케이션에 해당하는 번들의 실행 환경을 제공하는 표준 플랫폼이다. 하나의 번들은 0개 이상의 서비스들을 포함하는 서비스 집합이다. 그림 1과 같이 OSGi 프레임워크는 자바 가상 머쉰을 기반으로 작동되고 다수의 번들을 수행하기 위한 번들간의 공유 및 협력을 제공하고 있다 [2]. 한 대의 자바 가상 머쉰에서 동작하는 OSGi 프레임워크의 번들은 로컬 OSGi 서비스 레지스트리에 등록된 서비스만을 공유할 수 있다.



(그림 1) OSGi 프레임워크에서 번들의 서비스 공유



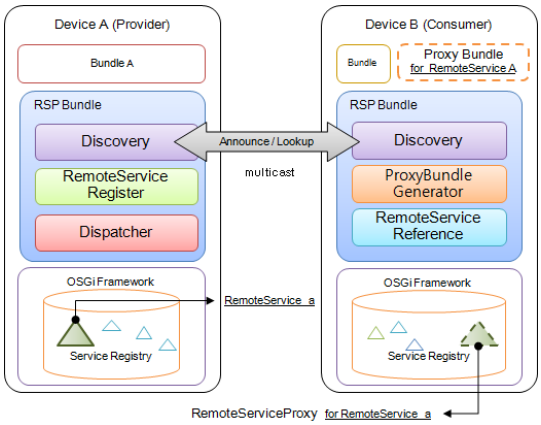
(그림 2) 분산 OSGi 프레임워크에서 장치들간의 상호운용 예

현재 컴퓨팅 환경은 분산, 이동, 유비쿼터스 컴퓨팅 환경으로 진화되고 있다. 유비쿼터스 컴퓨팅 환경에서는 고성능 컴퓨터, 데스크탑 PC, 이동 장비, 정보가전기기, 센서 등이 유무선 네트워크로 연결되고 서로 인터액션을 수행하여 사용자에게 서비스를 제공하는 구조로 점점 바뀌어 가고 있다. 홈 네트워크의 경우, 한대의 OSGi 프레임워크가 게이트웨이 역할을 하던 기존 홈 네트워크와 달리 그림 2와 같이 TV, 냉장고 등 가정내의 여러 정보가전기기 각각에 OSGi 프레임워크가 탑재되고 있으며, 데스크탑 PC, 외부의 오락 서버, 핸드폰 등에 OSGi 프레임워크가 탑재되어 컴퓨팅 환경이 구축되고 있다. 따라서, OSGi 프레임워크를 탑재한 머쉰들이 유무선으로 연결되어 구성된 유비쿼터스 컴퓨팅 환경에서는 분산되어 있는 OSGi 프레임워크 장치들을 서로 연동하여 원격의 OSGi 서비스들을 공유하여 상호운용할 수 있는 기술이 필요하다.

따라서 본 논문에서는 분산되어 있는 OSGi 프레임워크의 원격 OSGi 서비스를 로컬 OSGi 서비스처럼 호출하여 사용할 수 있도록 RMI 패러다임을 적용하여 원격 OSGi 상호운용 시스템을 설계하고 구현한다. RMI는 기존 분산 컴퓨팅 환경에서 원격 프로세스에 실행되고 있는 원격 객체의 메소드를 로컬 메소드처럼 호출 가능하게 하는 일종의 프로세스간 통신을 위한 미들웨어 기술이다. 통신 프로토콜과 메시지 구성을 위한 복잡한 처리는 RMI 미들웨어가 처리하므로 RMI를 사용하는 어플리케이션 개발자는 프로그램 쉽게 빨리 개발할 수 있다.

현재 OSGi는 다음과 같은 장점들을 가진다. 먼저 OSGi는 자바 기반의 컴포넌트 구조로 설계되었다. 따라서 번들이라는 여러개의 컴포넌트들이 조합되어서 하나의 어플리케이션을 구성하며, OSGi 프레임워크는 번들의 설치 및 삭제, 실행, 중지, 시작, 갱신을 관리한다. 그러므로 OSGi 프레임워크가 일단 설치되면, OSGi 어플리케이션의 배치 및 관리가 쉽다는 장점이 있다. 또한 OSGi 플랫폼에서 작성된 어플리케이션은 쉽게 다른 OSGi 플랫폼에서도 잘 동작하므로 호환성이 뛰어나며, OSGi 프레임워크의 재시작 없이도 번들의 설치 및 삭제, 갱신 등이 가능하다는 큰 장점들이 있다. 이러한 뛰어난 장점을 가진 OSGi 프레임워크에 RMI 패러다임을 적용한 원격 OSGi 서비스 상호운용 시스템을 구축하기 위해서는 다음과 같은 OSGi 특성을 고려해야 한다.

첫째, OSGi는 서비스 지향적인 아키텍처를 가지며,



(그림 3) 원격 OSGi 서비스 상호운용 시스템 구조

서비스 등록, 발견, 바인드(bind) 매커니즘을 사용하여 서비스를 접근한다는 특성이 있다. 따라서 본 논문에서는 서비스 지향적인 아키텍처를 가지는 OSGi 프레임워크에서 원격 OSGi 서비스 상호운용성을 제공하기 위하여 기존 OSGi에서 지원하는 로컬 서비스 레지스트리를 그대로 활용하면서, OSGi 서비스 레지스트리에 원격 서비스를 등록하고 발견할 수 있도록 확장한다. 구체적으로 OSGi의 서비스 속성을 기반으로 원격 서비스 속성을 정의하고, 로컬 OSGi 서비스 레지스트리에 등록하는 방안을 3.1절에서 제안한다. 또한 OSGi의 ServiceReference를 확장하여 원격서비스 레퍼런스를 획득하는 방안을 3.2절에서 제안한다.

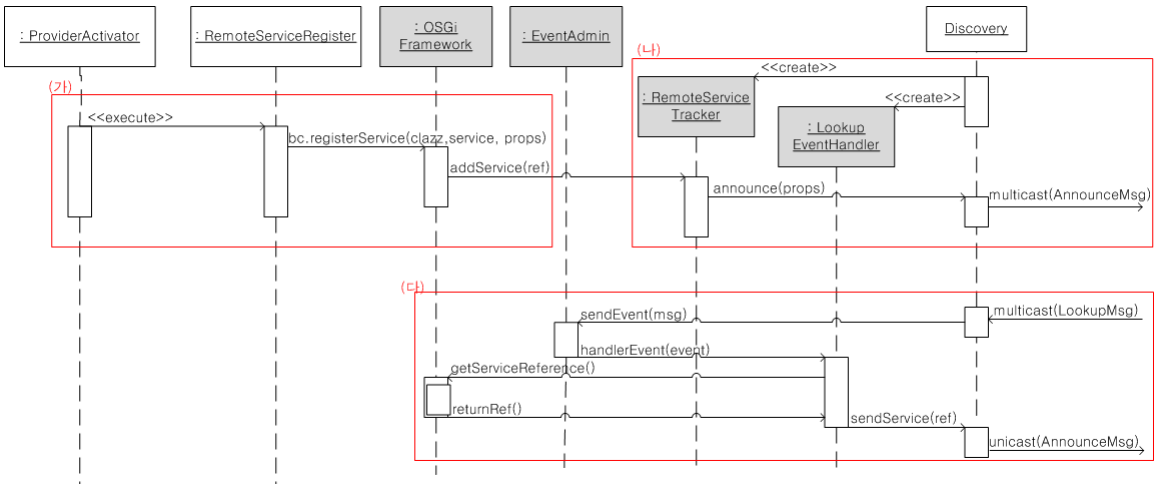
둘째, OSGi 번들의 속성은 동적으로 변한다. 즉, 번들의 설치 및 삭제, 갱신, 시작, 실행, 중지 등이 OSGi 프레임워크의 재시작 없이도 가능하므로, 번들의 상태가 동적으로 수시로 변한다. 따라서 원격 OSGi 서비스 상호운용 시스템에서도 이러한 동적인 특성을 지원하기 위하여 OSGi에서 표준서비스로 제공하는 ServiceTracker를 확장하여 RemoteServiceTracker를 설계하고, EventAdmin 서비스를 활용한 동적 원격서비스의 신뢰성을 제공하는 방안을 3.3절에서 제안한다.

또한 추가적으로 원격 OSGi 서비스를 로컬 OSGi 서비스처럼 호출하여 사용할 수 있는 위치 투명성을 제공하기 위한 Proxy 번들 및 Proxy 서비스 지원 방안을 3.4절에서 설명한다.

3. 원격 OSGi 서비스의 상호운용 시스템 설계 및 구현

그림 3은 원격 OSGi 서비스 상호운용 시스템의 전체 구조이다. 시스템은 Discovery, RemoteService Register, Dispatcher, ProxyBundleGenerator, Remote ServiceReference 서비스로 구성되며, 이러한 서비스들은 하나의 번들로서 구현된다. 본 논문에서는 원격 OSGi 서비스 상호운용 서비스들을 구현한 번들을 RSP(Remote Service Provider)라고 부른다. RemoteServiceRegister 서비스는 원격으로 공유할 서비스를 로컬 OSGi 서비스 레지스트리에 등록하는 서비스이다. Discovery 서비스는 원격 OSGi 서비스를 광고하거나 필요한 원격 OSGi 서비스를 찾아주는 기능을 제공한다. RemoteServiceReference 서비스는 OSGi 프레임워크의 번들이 원격에 있는 서비스를 사용하고자 할 때, 원격 서비스에 대한 서비스 레퍼런스를 획득해주는 서비스이다. Proxy BundleGenerator 서비스는 원격 OSGi 서비스에 대한 프록시(proxy) 서비스와 프록시 번들을 동적으로 생성해주는 서비스이다. 프록시 번들과 프록시 번들로 인해 원격 OSGi 서비스를 로컬 OSGi 서비스를 호출하듯 사용할 수 있게 된다. Dispatcher 서비스는 원격 서비스 요청을 받아들이고 실제 해당 서비스를 호출하는 서비스이다.

원격 OSGi 서비스를 사용하기까지 다음과 같은 과정을 거친다. 그림 3 디바이스 A의 OSGi 프레임워크에서 실행되는 번들 A는 원격 서비스 a를 RemoteServiceRegister 서비스를 이용하여 로컬 서비스 레지스트리에 등록한다. 디바이스 A의 Discovery 서비스는 등록된 원격 서비스 a를 광고한다. 디바이스 B의 번들 B가 원격 서비스 a를 사용하고자 한다면, 먼저 해당 원격 서비스의 원격 서비스 레퍼런스를 얻어야 한다. 따라서 번들 B는 RemoteServiceReference 서비스를 호출하여 원격



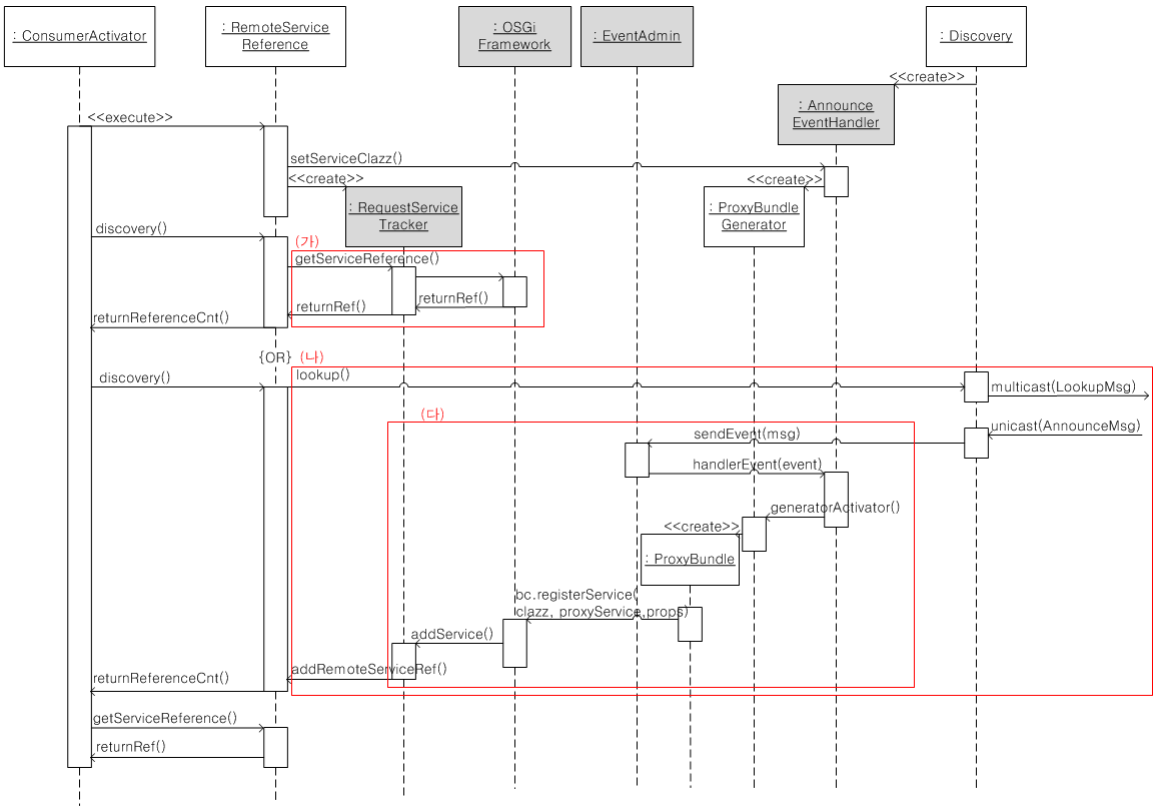
(그림 4) 원격 서비스 등록 및 광고, 룩업 요청에 대한 응답 시퀀스 다이어그램

서비스 레퍼런스를 얻는다. 그런데 디바이스 B의 OSGi 프레임워크에 원격 서비스 A에 대한 프락시 서비스가 등록되어 있지 않다면 원격 서비스 레퍼런스 획득에 실패하게 된다. 이러한 경우 RemoteServiceRefernce 서비스는 해당 프락시 서비스가 로컬 서비스 레지스트리에 등록되기를 기다린다. 이 때 디바이스 B의 Discovery 서비스가 원격 서비스 a에 대한 광고 메시지를 받게되면 자신이 기다리고 있는 원격 서비스인지를 검사한 후, ProxyBundleGenerator 서비스를 구동시킨다. ProxyBundleGenerator는 Discovery 서비스로부터 넘겨받은 원격 서비스 a에 대한 정보를 사용하여 프록시 번들과 프록시 서비스를 생성하고 구동시킨다. 원격 서비스 a에 대한 프록시 서비스가 디바이스 B의 OSGi 프레임워크에 등록되면 RemoteServiceReference 서비스는 이를 통보받게 되고, 원격 OSGi 서비스 a에 대한 원격 서비스 레퍼런스를 번들 B에 리턴한다. 번들 B는 리턴받은 원격 서비스 레퍼런스를 이용하여 원격 서비스 a를 호출하게 되고, 원격 서비스 요청은 프록시 서비스를 통해 Dispatcher 서비스에 전달된다. Dispatcher는 해당 서비스 a를 호출하여 실행하고 그 실행 결과를 다시 디바이스 B의 프록시 서비스에 리턴해 준다. 프록시 서비스는 번들 B에 결과를 리턴함으로써 원격 서비스 호출을 완료하게 된다.

3.1 원격 서비스 속성 정의와 등록

OSGi에는 어플리케이션에 해당하는 번들이 있다. 하나의 번들은 0개 이상의 서비스들로 구성되고, 번들이 시작될 때 서비스들의 인터페이스 이름과 속성을 서비스 레지스트리에 등록하게 된다. 번들이 정지되면 서비스 레지스트리에 등록된 해당 번들의 모든 서비스들은 삭제된다. 원격서비스는 로컬 OSGi 프레임워크내에서 공유될 수 있어야 하고, 또한 원격 OSGi 프레임워크에서도 공유돼야 하므로, OSGi 서비스 레지스트리에 등록될 때 서비스가 원격으로 서비스가 가능하다는 것을 알려야 한다.

원격 OSGi 서비스 상호운영 시스템에서는 이를 위하여 OSGi에서 제공하는 서비스의 속성을 사용한다. OSGi에서는 서비스의 속성을 '속성이름=값'으로 정의한다. 원격 서비스임을 알리기 위해 'service.remote=true' 라는 새로운 속성이름과 속성값을 정의하였다. RSP에서 원격 서비스를 로컬 OSGi 레지스트리에 등록하는 서비스는 RemoteServiceRegister이며, 그림 4의 (가)는 서비



(그림 5) RemoteServiceReference 서비스의 시퀀스 다이어그램

스 제공자가 RemoteServiceRegister 서비스를 사용하여 원격 서비스를 등록하는 시퀀스 다이어그램을 나타낸다.

3.2 원격 서비스 레퍼런스

3.2.1 서비스 레퍼런스를 확장한 원격 서비스 레퍼런스

로컬에 등록된 서비스들을 다른 번들에서 사용하고자 할 때, 먼저 OSGi 서비스 레지스트리를 검색하여 해당 서비스의 ServiceReference를 획득해야 한다. 마찬가지로 원격 OSGi 서비스를 사용하고자 할 때, 먼저 OSGi 서비스 레지스트리를 검색하여 해당 서비스의 원격 ServiceReference를 획득하도록 설계하였다. 따라서 원격 서비스 레퍼런스는 로컬 ServiceReference를 확장한 RemoteServiceReference를 설계한다. RemoteServiceReference 서비스를 통해 리턴되는 RemoteServiceReference는 OSGi의 ServiceReference와 동일하게 서비스 아이디, 서비스 오브젝트와 같은 정보를 포함하며, 추가적으로 원격 서비스에 대한 IP 주소, 포트번호를 포함한다.

3.2.2 원격 서비스 레퍼런스 획득

OSGi 프레임워크의 번들이 원격에 있는 OSGi 서비스를 사용하고자 할 때, 먼저 RSP의 RemoteServiceReference 서비스를 사용하여 원격 서비스에 대한 ServiceReference를 획득해야 한다.

그림 5의 (가)와 같이 RemoteServiceReference 서비스는 원격 서비스에 대한 ServiceReference를 획득하기 위하여 원격 서비스에 대한 프록시 서

비스가 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, 그림 5의 (나)와 같이 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.

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 ProxyBundle Generator을 생성한다.

3.3 원격 서비스의 발견

원격 OSGi 서비스의 상호운용을 지원하기 위해서는 분산 OSGi 프레임워크 환경에서 사용하고자 하는 원격 OSGi 서비스를 발견할 수 있어야 한다. 이를 위하여 본 논문에서는 원격서비스가 등록되었을 때 등록된 원격 서비스를 분산 OSGi 프레임워크 환경에 광고하는 기능을 지원하고, 사용하고자 하는 원격 서비스를 찾을 수 있도록 룩업(lookup) 기능을 지원하는 Discovery 서비스를 설계하였다. RSP의 Discovery 서비스는 원격서비스 광고와 룩업을 위하여 멀티캐스트 통신 채널(multicast communication channel)과 유니캐스트 통신 채널(unicast, communication channel)을 가지며, OSGi의 ServiceTracker, EventAdmin 서비스를 활용하여 설계하였다. 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 구조로 설계하였다.

3.3.1 원격 서비스 광고

OSGi 서비스 플랫폼에서 번들들은 언제든지 설치되거나 삭제될 수 있고, 또한 언제든지 번들이 시작되거나 종료될 수 있는 속성이 있기 때문에 OSGi 서비스들은 매우 동적인 특성이 있다. 따라서 번들이 실행되면 등록된 새로운 서비스를 사용할 수 있게 되고, 번들의 상태가 변하면 번들에 포함된 모든 서비스들도 사용이 불가능해질 수 있다. 분산 OSGi 프레임워크 환경에서 원격 서비스 또한 이러한 동적인 특성을 가진다. OSGi 프레임워크에서는 ServiceTracker라는 표준 서비스를 제공함으로써 동적인 로컬 서비스의 속성을 추적할 수 있도록 지원하고 있다. 본 논문에서는 동적으로 생성되는 원격 서비스의 광고를 위하여 원격서비스 등록을 추적할 수 있도록 OSGi의 표준 서비스인 ServiceTracker를 확장한 Remote ServiceTracker를 설계하였다.

그림 4의 (나)와 같이 RSP의 Discovery 서비스는 RemoteServiceTracker를 사용하여 로컬 OSGi 서비스 레지스트리에 원격 OSGi 서비스가 등록되는지를 모니터링하다가, 원격 서비스가 등록되었을 때 이를 분산 OSGi 프레임워크 환경에 광고하는 기능을 제공하여 해당 원격 서비스가 등록되기를 기다리는 클라이언트가 원격 서비스를 발견할 수 있게 한다. 뿐만 아니라, RemoteService Tracker는 원격 서비스의 삭제, 갱신을 실시간으로 모니터링하여 원격 서비스의 상태변화를 광고

하는 기능을 가진다. 원격 OSGi 서비스의 클라이언트측에 해당하는 Discovery 서비스는 원격 서비스의 삭제, 갱신 메시지를 받아 이벤트를 발생하여 해당 원격 서비스에 대응하는 ProxyBundle의 라이프 사이클을 관리한다. 원격 서비스의 등록, 삭제, 갱신 광고를 할 때, 네트워크의 효율적 사용을 위해 1:N 통신을 제공하는 멀티캐스트 통신 채널을 사용한다. 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. 이 때, IP 주소는 원격 서비스를 제공하는 OSGi 프레임워크 머신의 IP 주소이고, 포트 번호는 Dispatcher가 원격 서비스 요청을 받기 위해 사용하는 포트 번호이다. 멀티캐스트 통신 채널로 새로 등록된 원격 서비스의 광고 메시지를 받은 피어의 Discovery 서비스는 EventAdmin 서비스를 사용하여 원격 서비스가 등록되었다는 것을 알리는 이벤트를 발생시킨다. 발생되는 이벤트는 해당 원격서비스 정보를 포함하고 있다. 이벤트 핸들러는 광고된 원격서비스가 등록되기를 기다리고 있던 원격서비스가 맞을 때 ProxyBundleGenerator 서비스를 구동시킨다.

3.3.2 원격 서비스 룩업

원격 OSGi 프레임워크가 원격 서비스를 제공 가능한 상태일 때, 로컬 OSGi 프레임워크에서 원격 서비스를 사용하고자 하는 번들이 구동되어 Discovery 서비스에게 원격 서비스를 요청(lookup)하는 경우가 발생한다. 원격 OSGi 서비스를 사용하고자 하는 번들이 RemoteServiceReference 서비스에게 해당 서비스를 요청하면, 그림 5의 (가)와 같이RemoteServiceReference는 우선 로컬 OSGi 서비스 레지스트리에서 해당 원격 서비스의 프록시 서비스를 찾고, 프록시 서비스를 발견하지 못했을 경우 그림 5의 (나)와 같이 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, 원격 서비스를 가지고 있는 피어가 룩업 응답 메시지 낼 때, 메시지 전송의 효율을 위하여 이 유니캐스트 채널 정보를 사용한다.

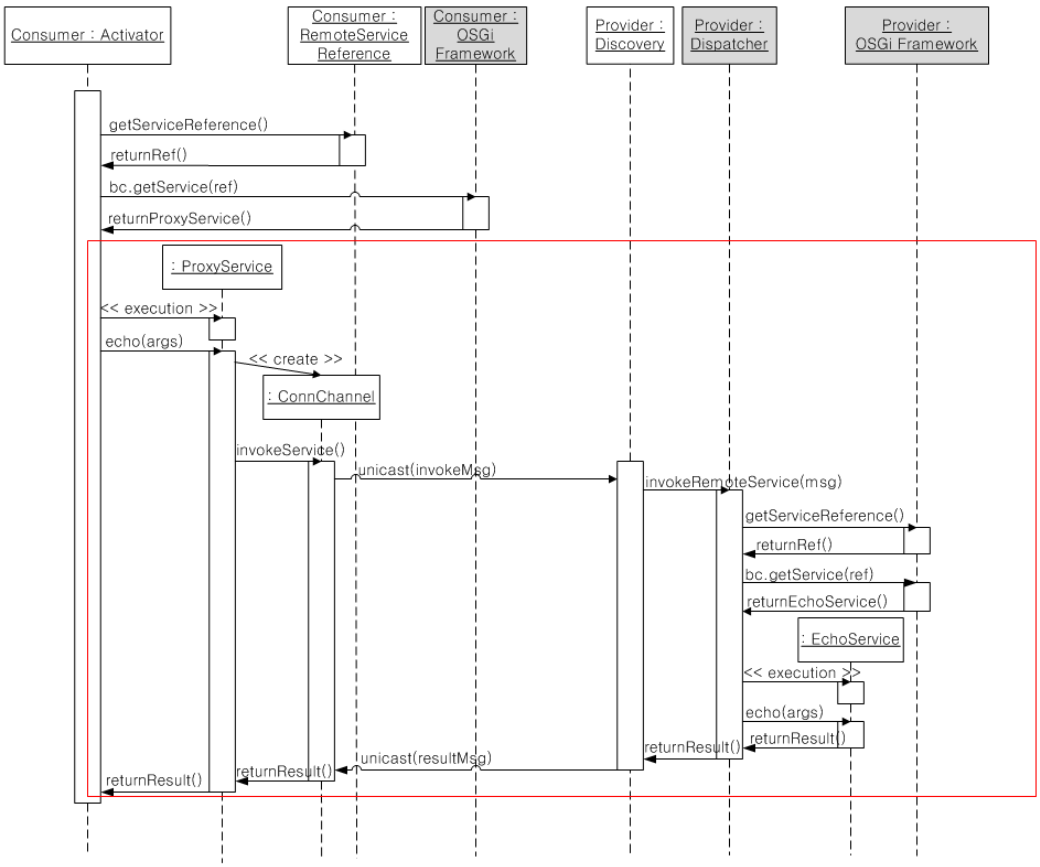
원격 서비스 룩업 요청 메시지를 받은 피어의 Discovery 서비스는 그림 4 (나)와 같이 요청 서비스 정보를 포함하는 룩업 이벤트를 발생한다. 발생한 룩업 이벤트를 처리하는 RemoteService Register가 있다면, RemoteServiceRegister에 등록되어있는 요청 서비스 정보인 서비스 이름과 서비스 속성를 포함한 룩업 응답 메시지를 TCP 통신을 사용하여 응답한다. 룩업 응답 메시지의 목적지는 룩업 요청 메시지에 포함되어 있는 TCP 통신 정보를 사용하여 룩업 서비스 요청자에게 전송한다.

3.4 투명성을 제공하는 원격 서비스 호출

3.4.1 프록시 번들과 프록시 서비스의 동적 생성

어플리케이션 개발자는 원격 서비스의 호출이 로컬 서비스 호출과 같은 방법으로 이루어지기를 요구한다. 이러한 투명성으로 인해 개발자는 어플리케이션 로직 개발에 더 집중할 수 있기 때문이다. 따라서 원격 OSGi 서비스 상호운용 시스템에서 원격 OSGi 서비스의 호출은 로컬 OSGi 서비스 호출과 같은 방법으로 이루어지도록 설계하였다. OSGi에서 서비스는 번들이라는 소프트웨어 단위에 포함된다는 특징이 있으므로, 본 논문에서는 이러한 투명성을 제공하기 위하여 원격 OSGi 서비스를 로컬 OSGi 서비스처럼 동작가능하게 하는 프록시 서비스를 설계하고, 프록시 서비스를 포함하는 프록시 번들을 설계하였다.

RSP의 ProxyBundleGenerator는 원격 OSGi 서비스에 대한 프록시 서비스와 이 프록시 서비스를 포함하는 ProxyBundle을 동적으로 생성시키는 서



(그림 6) 원격 서비스 호출 시퀀스 다이어그램

비스이다. ProxyBundleGenerator가 생성한 Proxy Bundle은 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. 그림 5의 (다)와 같이 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 RemoteService Proxy를 생성한다. ProxyBundle이 OSGi 프레임워크에서 구동될 때, Activator 클래스는 RemoteServiceProxy를 로컬 OSGi 서비스 레지스트리에 실제 서비스와 같이 등록한다.

OSGi에서 번들의 설치, 구동, 삭제, 갱신과 같은 번들 라이프 사이클은 OSGi 프레임워크가 관리한다. ProxyBundleGenerator가 생성한 Proxy Bundle의 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. 본 논문에서는 ProxyBundleGenerator가 원격 OSGi 서비스에 해당하는 ProxyBundle의 코드를 동적으로 생성하도록 ASM [6] 라이브러리

를 사용하였다. ASM은 자바 클래스들을 동적으로 생성하고 조작할 수 있도록 라이브러리를 제공하는 도구이다.

3.4.2 원격 메소드 호출

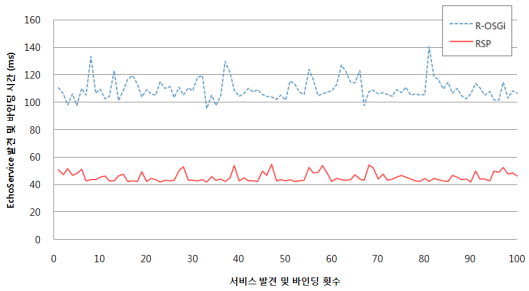
RSP의 Dispatcher는 원격 서비스를 제공하는 서버측 OSGi 프레임워크에서 클라이언트의 원격 서비스 요청을 받아들이고 처리하는 데몬 서비스이다. RSP 번들이 구동되면 Dispatcher 서비스가 실행되어 원격 서비스에 대한 클라이언트 호출 요청을 기다리게 된다.

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 TCP 통신을 한다.

그림 6과 같이 클라이언트측 RemoteService Proxy는 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.

4. 성능 평가

본 논문에서 제안한 원격 OSGi 서비스 상호운용 시스템의 성능을 평가하고, 6장 관련 연구에서 언급하는 R-OSGi의 성능과 비교 분석하였다. R-OSGi는 분산 OSGi 프레임워크들의 서비스를 상호운용하기 위해 개발자에게 RMI 매커니즘을 제공하며 중앙 집중적인 서비스 레지스트리를 사용하는 미들웨어이다. DPWS(Devcie Profile for Web Services), JXTA 또는 Web Services 같은 기존의 분산

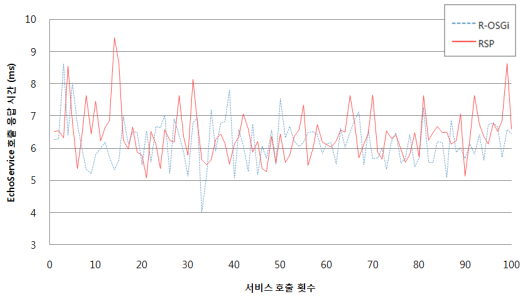


(그림 7) RSP와 R-OSGi의 서비스 발견 및 바인딩 시간 비교

미들웨어를 이용하는 기술들과 비교했을 때, 본 연구에서 개발한 분산 OSGi 상호운용 기술과 같이 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 3.6절에서 설명한 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 측정하였



(그림 8) RSP와 R-OSGi의 서비스 호출 시간 비교

다. 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 때를 각각 그 성능을 측정하였다.

그림 7은 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 45.6ms이고, R-OSGi의 평균 시간은 약 109.6ms로 측정되었다. RSP의 서비스 발견 및 바인딩 시간은 R-OSGi보다 약 2배 이상 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 훨씬 빠르게 측정된 것이다. 그리고 R-OSGi는 서비스 발견을 위하여 SLP 기술을 접목시켰지만 RSP는 서비스 발견을 위하여 OSGi 자체 기술인 ServiceTracker와 EventAdmin 서비스를 이용하여 OSGi 프레임워크 안에서 더욱 빠른 서비스 발견을 하는 것이다.

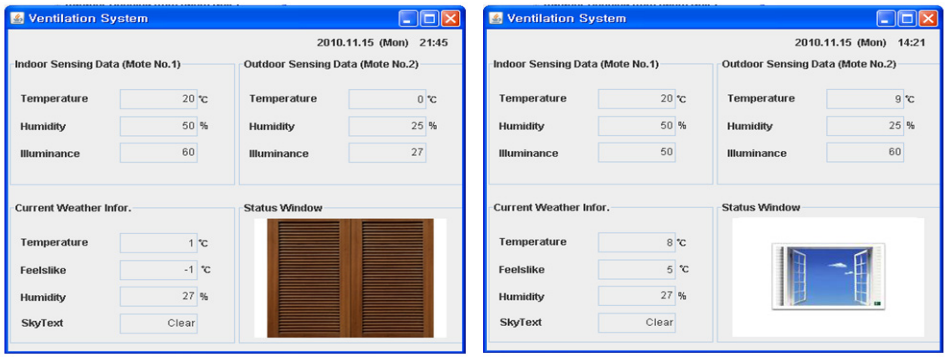
그림 8은 EchoService의 아규먼트가 null일 때, RSP와 R-OSGi의 호출 시간을 100회에 걸쳐 측정한 실험결과를 나타낸다. RSP의 평균 EchoService 호출 시간은 평균 약 14.4ms이고, R-OSGi는 약 평균14.9ms로 측정되었다. EchoService 호출 시간은 RSP와 R-OSGi 모두가 프록시를 이용하여 원격 서비스를 호출하기 때문에 거의 비슷한 결과를 얻었다. 100회의 실험결과를 통해 RSP와 R-OSGi가 원격 아규먼트가 null인 EchoService를 호출하여 실행 결과를 돌려받는데 걸린 총 소요 평균 시간은 [표 1]과 같다. RSP는 평균 약 51.9ms, R-OSGi는 약 115.8ms의 시간이 걸렸으므로 RSP가 R-OSGi보다 약 2배 빠른 성능을 보인다. [표 1]에는 EchoService의 아규먼트가 512byte일 때와 1024byte일 때의 각각의 성능을 나타내고 있다.

(표 3) 원격 서비스를 이용하기 위한 소요 시간 (ms)

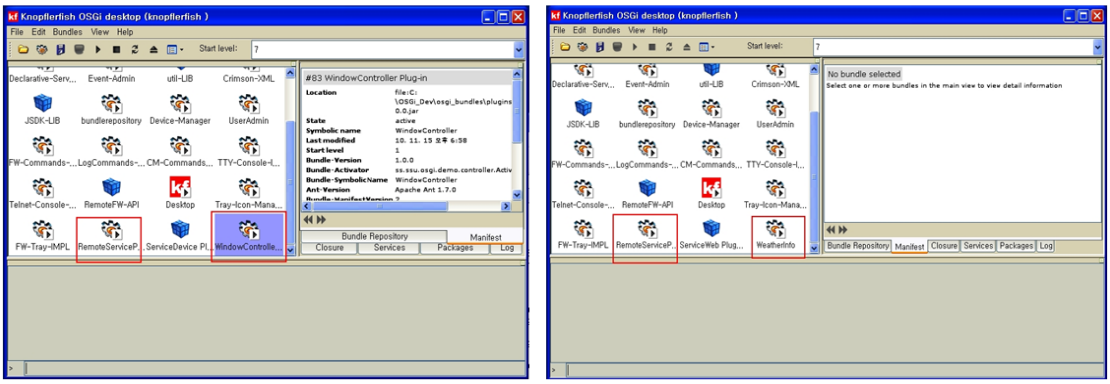
구분	서비스 발견 및 바인딩		서비스 호출		총 소요 시간	
	R-OSGi	RSP	R-OSGi	RSP	R-OSGi	RSP
null	109.6	45.6	6.2	6.4	115.8	51.9
512 byte	113.1	46.1	6.6	6.4	119.7	52.5
1024 byte	113.5	45.8	6.6	6.8	120.3	52.6

5. 어플리케이션

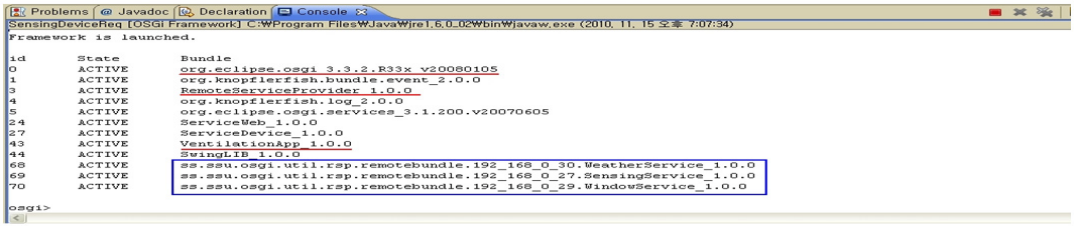
본 논문에서 제안한 원격 OSGi 서비스 상호운용을 지원하는 RSP 번들의 응용으로 유비쿼터스 홈 환경의 환기(ventilation) 어플리케이션을 개발하였다. 그림 9는 OSGi 환기 어플리케이션의 실행화면이다. 환기 어플리케이션은 자바 스윙 번들



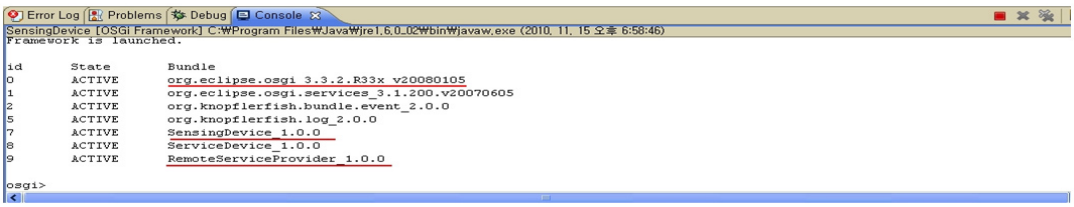
(그림 9) 센싱 정보에 따른 환기 어플리케이션의 동작



(그림 10) Knopflerfish OSGi 프레임워크에서 Window 서비스와 Weather 서비스의 실행



(a) Ventilation 어플리케이션과 세 개의 원격 서비스



(b)SensingDevice 서비스

(그림 11) Equinox OSGi 프레임워크에서 실행중인 Ventilation 어플리케이션과 SensingDevice 서비스

을 사용하여 개발하였고, OSGi 서비스로 동작한다. 환기 어플리케이션은 실내, 실외, 날씨 정보를 끊임없이 센싱 및 모니터링하여 시간대, 실내외 온도, 습도 광도의 차이, 날씨 정보를 비교하여, 창문을 열고 닫아 자동으로 환기시키는 어플리케이션이다. 한백전자 ZigbeX ver 1.4 모델의 센서를 사용하여 온도, 습도, 광도를 측정하였고, 날씨정보는 야후에서 제공하는 오픈 API를 사용하여 날씨 정보를 수집하였다. 또한 개발한 RSP 번들이 서로 다른 OSGi 프레임워크에서 도 동작한다는 것을 보이기 위해, 그림 10과 같이 창문을 제어하는 Window 서비스와 날씨정보를 모니터링하는 Weather 서비스는 Knopflerfish OSGi 프레임워크에서 실행시키고, 그림 11과 같이 날씨, 온도, 광도를 센싱하는 SensingDevice 서비스와 환기 어플리케이션은 Equinox OSGi 프레임워크에서 실행시켜 RSP의 멀티벤더 상호운용성을 보였다. 그림 10과 그림 11의 Knopflerfish와 Equinox 프레임워크에는 각각 원격 서비스 지원을 위한 RSP 번들이 실행되고 있다. 또한, 그림 11의 (a)는 환기 어플리케이션이 Weather, Sensing, Window 서비스를 원격으로 사용하고 있음을 보여준다.

6. 관련연구

분산 OSGi 프레임워크에서 원격 OSGi 서비스의 상호운용을 지원하기 위한 여러 연구들이 진행되어 왔다. 현재 가장 많이 사용되는 OSGi 4.1 릴리즈 스펙에서는 UPnP 서비스와의 상호연동을 위한 OSGi 서비스 스펙을 제공하고 있지만, 분산되어 있는 OSGi 프레임워크 장치들을 서로 연동하여 원격 OSGi 서비스들을 상호운용할 수 있는 방안을 제공하지 못하고 있다. 2009년 9월에 릴리즈된 OSGi 4.2 에서는 분산 OSGi의 상호운용을 위한 기술을 스펙에서 정의하고 있지만 OSGi 기반이 아닌 서비스와 클라이언트까지 광범위하게 포괄할 수 있도록 스펙에서 정의하기에 대부분의 벤더들은 아직 이를 지원하고 있지 않은 실정이다.

원격 OSGi 서비스를 상호운용하기 위한 기술로는 DPWS-OSGi [4], p2pcomp [7], 스마트 아키텍처 [8], R-OSGi [9]와 같은 연구들이 있다. 이러한 연구들은 각각 기존의 분산 기술인 P2P 기반의 JXTA(Juxtapose) [10], Web Services [11], SLP (Service Location Protocol) [12], DPWS(Devcie Profile for Web Services) [4]와 같은 기술들을 활용하는 기술들이다. DPWS-OSGi, p2pcomp, 스마트 아키텍처의 경우, OSGi 프레임워크에 각각 DPWS, JXTA와 Web Services를 번들로 실행시켜야 한다. OSGi 프레임워크는 컴퓨팅 자원이 부족한 이동, 임베디드 장비에도 많이 탑재되기 때문에 이러한 미들웨어를 구현한 번들을 실행시킬 때 컴퓨팅 성능의 문제가 발생할 수 있다. 또한 스마트 아키텍처에서는 MASML (the Mobile Agent Specification Markup Language)로 기술된 작업 정의 문서를 OSGi 서비스로 변환해야 하는 서비스 변환과정이 필요하다는 단점이 있다. R-OSGi는 중앙집중적인 서비스 레지스트리를 사용하는 SLP를 사용함으로써 OSGi 내부의 서비스 레지스트리와는 별도로 중앙 서비스 레지스트리를 유지해야 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.

본 연구에서는 제안하는 원격 OSGi 서비스 상호운용 기술은 OSGi에서 정의한 서비스 스펙을 따르므로 서비스를 변환하는 과정이 부가적으로 필요하지 않다. OSGi 프레임워크에 기존 분산 기술을 번들로 구현할 필요없이 OSGi 표준 기술은 확장하여 사용하기 때문에 기존 분산 미들웨어를 활용하는 기술들과 비교했을 때 컴퓨팅 자원소모가 적다는 특징을 가진다. 또한 P2P 방식의 서비스 발견 기술을 통해 네트워크 병목현상과 SPOF 문제가 발생하지 않도록 하였고, 원격 서비스의 동적인 상태변화를 반영함으로써 원격 서비스에 대한 신뢰성을 보장한다. 또한 일반적인 프로그래

밍 패러다임인 RMI 방식으로 분산 OSGi 서비스를 개발할 수 있게 함으로서 투명성 있는 원격 OSGi 서비스 개발이 가능하다는 장점이 있다.

7. 결 론

유비쿼터스 환경의 OSGi는 홈 네트워크의 표준으로 인식되던 과거와는 달리 디지털 이동 전화, 차량, 텔레메틱스, 임베디드 가전, 가정용 게이트웨이, 산업용 컴퓨터, 데스크탑 컴퓨터, 고성능 서버에 이르기까기 그 적용범위가 확대되고 있다. 따라서 다양한 장비에 탑재된 OSGi 프레임워크의 서비스들을 상호운용할 수 있는 기술이 필요하게 되었다.

본 논문에서는 분산 OSGi 프레임워크에서 원격 서비스의 상호운용을 가능하도록 분산컴퓨팅에서 대표적인 프로세스간 통신을 위한 미들웨어 기술인 RMI 패러다임을 적용한 원격 OSGi 서비스 상호운영 방안을 제안한다. 제안하는 원격 OSGi 서비스 상호운용 방안은 OSGi 표준 기술을 활용 및 확장하여 서비스지향적인 OSGi 아키텍처에 부합하는 원격서비스의 등록 및 발견, 접근 방법을 제공하며, 동적으로 변하는 원격 서비스들의 속성을 일관성 있게 유지하여 원격서비스에 대한 신뢰성을 지원한다. 또한 동적으로 프락시 번들 및 프락시 서비스를 생성함으로써 원격 OSGi 서비스의 위치 투명성을 지원하는 특징을 가진다. 또한 P2P방식의 서비스 발견 메커니즘을 사용하여 중앙집중적인 서비스 레지스트리 방식의 문제점을 해결하였다. OSGi 표준 기술을 확장하여 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 상호운용성을 지원하는 확장된 분산 OSGi 기술의 연구가 추가적으로 더 필요하다.

참 고 문 헌

[1] Vijakumar Krishnaswamy, Dan Walther, Sumeer Bhola, Ethendranath Bommajah, George Riley, Brad Topol, Mustaque Ahamad, "Efficient Implementation of Java Remote Method Invocation", 4th USENIX Conference on Object-Oriented Technology and Systems, 1998.

[2] OSGi Alliance, "About the OSGi Service Platform", June 2007.

[3] Jingang Zhou, Dazhe Zhao, Yong Ji, Jiren Liu, "Examining OSGi from an Ideal Enterprise Software Component Model", IEEE International Conference on Software Engineering and Service Sciences, 2010.

[4] O. Dohndrof, J. Kruger, H. Krumm, C. Fiehe, A. Litvina, I. Luck, F. Stewing, "Toward the Web of Things: Using DPWS to bridge isolated OSGi platforms", Pervasive Computing and Communication Workshops, 2010.

[5] Chang Cheng, Yue Suo, Yu Chen, Yuanchun Shi, Weikang Yang, "SSCP: An OSGi-based Communication Portal for Smart Space", Joint Conference on Pervasive Computing(JCPC), 2009.

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

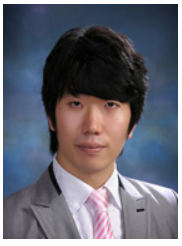
[7] A Ferscha, M Hechinger, R Mayrhofer, R Oberhauser, "A Light-Weight Component Model for Peer-to-Peer Applications", Distributed Computing Systems Workshops, 2004.

[8] 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.
[9] J. S. Rellermeyer, G. Alonso, "Service Everywhere: OSGi in Distributed nvironments," In EclipseCon, 2007.
[10] JXTA, http://jxta.dev.java.net
[11] Web Services, http://www.w3.org/standards/webofservices/, W3C.
[12] Erik guttman, "Service Location Protocol : Automatic Discovert of IP Network Services", IEEE Internet Computing Magazine, pp.71-80, July-August, 1999.

◐ 저 자 소 개 ◑

김 은 희
1989년 숭실대학교 전자계산학과 졸업(학사)
1996년 숭실대학교 대학원 컴퓨터학과 졸업(석사)
2006년 숭실대학교 대학원 컴퓨터학과 졸업(박사)
2007~2009년 숭실대학교 정보미디어기술연구소 전임연구원
2009~현재 숭실대학교 지능형로봇연구소 연구원
관심분야 : 병렬/분산처리, 유비쿼터스 컴퓨팅, RFID
E-mail : ehkim@ss.ssu.ac.kr



윤 기 현
2007년 목원대학교 컴퓨터공학과 졸업(학사)
2009년 숭실대학교 대학원 컴퓨터학과 졸업(석사)
관심분야 : 유비쿼터스, 클라우드컴퓨팅
E-mail : yunkiv@gmail.com

최 재 영
1984년 서울대학교 제어계측공학과 졸업(학사)
1986년 미국 남가주대학교 전기전자공학과 졸업(석사)
1991년 미국 코넬대학교 전기전자공학부 졸업(박사)
1992~1994년 미국 국립 오크리지연구소 연구원
1994~1995년 미국 테네시 주립대학교 연구교수
1995~현재 숭실대학교 컴퓨터학부 교수
관심분야 : 분산시스템, HPC, 시스템소프트웨어
E-mail : choi@ssu.ac.kr