한국컴퓨터정보학회 동계학술대회 논문집 제19권 제1호 (2011. 1)

OSGi 서비스의 원격 공유를 위한 기술 설계 및 구현

백경윤°, 윤기현*, 김은회*, 최재영*
°*숭실대학교 컴퓨터학부
e-mail: kybaek@ss.ssu.ac.kr, kihyun@ss.ssu.ac.kr, ehkim@ss.ssu.ac.kr, choi@ssu.ac.kr

Design and Implementation for the Remote sharing of the OSGi services

Kyoungyun Baek°, Kihyun Yun*, Eunhoe Kim*, Jaeyoung Choi*
°*School of Computing, Soongsil University

● 요 약 ●

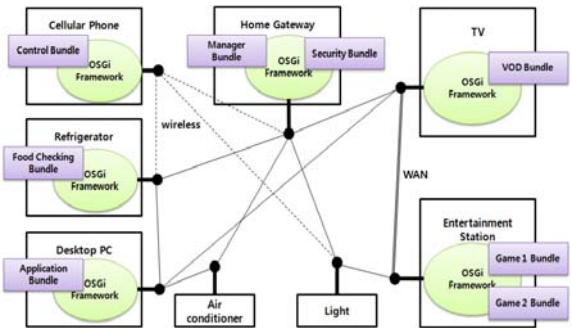
현재 컴퓨팅 환경은 분산, 이동, 유비쿼터스 컴퓨팅 환경으로 진화되고 있다. OSGi는 유비쿼터스 환경에서 디지털 이동 전화, 차량, 임베디드 가전, 가정용 게이트웨이, 데스크탑 컴퓨터, 고성능 서버에 이르기까지 그 적용범위가 확대되고 있다. 따라서 서로 다른 장비에 탑재된 OSGi 프레임워크의 서비스들을 서로 연동하여 원격 OSGi 서비스들을 공유하여 상호 운용할 수 있는 기술이 필요하게 되었다. 본 논문에서는 분산 OSGi 프레임워크에서 원격 서비스의 상호운용을 지원하기 위하여 분산 미들웨어 기술인 RMI 패러다임을 적용한 원격 OSGi 서비스 상호운영 방안을 제안한다. 제안하는 원격 OSGi 서비스 상호운용 방안은 OSGi 표준 기술을 활용 및 확장하여 OSGi 아키텍처에 부합하는 원격서비스의 등록 및 발견, 접근 방법을 제공한다. 또한 동적으로 프락시 번들 및 프락시 서비스를 생성함으로써 원격 OSGi 서비스의 위치 투명성을 지원하는 특징을 가진다.

키워드: OSGi(Open Service Gateway Initiative), RMI(Remote Method Invocation)

I. 서 론

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 상호작용을 수행하여 사용자에게 서비스를 제공하는 구조로 점점 바뀌어 가고 있다.

유비쿼터스 환경의 자바 가상 머쉰이 동작하는 다양한 디바이스에서 서비스의 공유와 배포를 위한 미들웨어로 크게 각광받는 기술은 OSGi이다. 2000년 OSGi Alliance에서 발표한 OSGi는 현재 홈 네트워크 뿐만 아니라, 디지털 이동 전화, 차량, 텔레메틱스, 임베디드 가전, 가정용 게이트웨이, 산업용 컴퓨터, 데스크탑 컴퓨터, 고성능 서버에 이르기까지 그 적용범위가 확대되고 있다. 이 중 홈 네트워크의 경우, 한 대의 OSGi 프레임워크가 게이트웨이 역할을 하던 기존 홈 네트워크와 달리 [그림 1]과 같이 TV, 냉장고 등 가정 내의 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 기술의 필요성이 점차 증대되는 상황이다.



[그림 1] 분산 OSGi 프레임워크에서 장치들간의 상호운용 예

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

따라서 본 논문에서는 분산 OSGi 프레임워크에서 원격 서비스의 상호운용이 가능하도록 대표적인 분산 미들웨어 기술인 RMI

를 적용하여 원격 OSGi 서비스 상호운영 방안을 제안한다.

II. 관련 연구

분산 OSGi 프레임워크에서 원격 OSGi 서비스의 상호운용을 지원하기 위한 여러 연구들이 진행되어 왔다. 그 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) [5], Web Services [6], SLP(Service Location Protocol) [7], DPWS(Device Profile for Web Services)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 MASML로 기술된 작업 정의 문서를 OSGi 서비스로 변환해야 하는 서비스 변환과정이 필요하다는 단점이 있다. R-OSGi는 중앙 집중적인 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 기술을 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(Single Point of Failure) 문제가 발생하지 않도록 하였고, 원격 서비스의 동적인 상태변화를 반영함으로써 원격 서비스에 대한 신뢰성을 보장한다. 추가적으로 일반적인 프로그래밍 패러다임인 RMI 방식으로 분산 OSGi 서비스를 개발가능하게 함으로써 투명성 있는 원격 OSGi 서비스 개발이 가능하다는 장점이 있다.

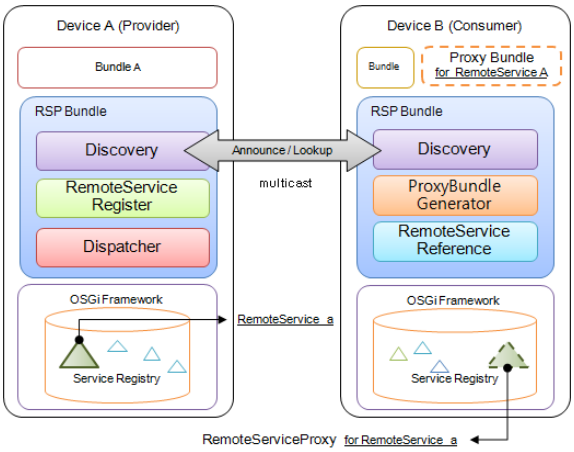
III. 원격 OSGi 서비스의 상호운용 시스템 설계 및 구현

1. 원격 OSGi 서비스의 시스템 구조

[그림 2]는 원격 OSGi 서비스 상호운용 시스템의 전체 구조이다. 시스템은 RemoteServiceRegister, Dispatcher, Discovery, ProxyBundleGenerator, RemoteServiceReference 서비스로 구성되며, 이 서비스들은 하나의 번들로 구현된다. 본 논문에서는 원격 OSGi 서비스 상호운용 서비스들을 구현한 번들을 RSP(Remote Service Provider)라 부른다.

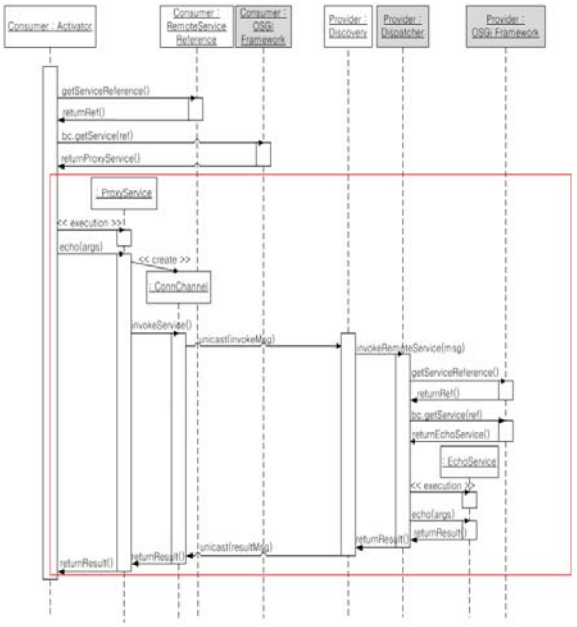
2. 원격 OSGi 서비스의 상호운용

RemoteServiceRegister 서비스는 원격으로 공유할 서비스를 로컬 서비스 레지스트리에 등록하는 역할을 한다. RemoteServiceReference 서비스는 원격에 있는 서비스를 사용하고자 할 때, 원격 서비스에 대한 서비스 레퍼런스를 획득해주는 서비스이다. ProxyBundleGenerator 서비스는 원격 서비스에 대한 프록시 서비스와 번들을 동적으로 생성해주는 역할을 한다. ProxyBundleGenerator 서비스를 통해 생성된 프록시 번들과 프록시 서비스를 통해 원격 OSGi의 서비스를 로컬서비스를 사용하듯 사용할 수 있다. Dispatcher 서비스는 원격 서비스 요청을 받아들이고 실제 해당 서비스를 호출하는 서비스이다. Discovery 서비스는 원격 OSGi 서비스를 광고하거나 찾아주는 기능을 제공한다.



[그림 2] 원격 OSGi 서비스 상호운용 시스템 구조

RSP를 사용하기 위해서는 다음과 같은 과정을 거친다. 디바이스 A의 OSGi 프레임워크에서 실행되는 번들 A는 원격 서비스 Hello를 RemoteService Register 서비스를 이용하여 로컬 서비스 레지스트리에 등록한다. 디바이스 A의 Discovery 서비스는 등록된 원격 서비스 Hello를 광고한다. 디바이스 B의 번들 B가 원격 서비스 Hello를 사용하고자 한다면, 먼저 해당 원격 서비스의 원격 서비스 레퍼런스를 얻어야 한다. 따라서 번들 B는 RemoteServiceReference 서비스를 호출하여 원격 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 RemoteServiceReference 서비스는 해당 프록시 서비스가 로컬 서비스 레지스트리에 등록되기를 기다린다. 이 때 디바이스 B의 Discovery 서비스가 원격 서비스 Hello에 대한 광고 메시지를 받게 되면 자신이 기다리고 있는 원격 서비스인지를 검사한 후, ProxyBundleGenerator 서비스를 구동시킨다. ProxyBundleGenerator는 Discovery서비스로부터 넘겨받은 원격 서비스 Hello에 대한 정보를 사용하여 프록시 번들과 프록시 서비스를 생성하고 구동시킨다. 원격 서비스 Hello에 대한 프록시 서비스가 디바이스 B의 OSGi 프레임워크에 등록되면



[그림 3] 원격 서비스 호출 시퀀스 다이어그램

RemoteServiceReference 서비스는 이를 통보받게 되고, 원격 OSGi 서비스 Hello에 대한 원격 서비스 레퍼런스를 번들 B에 리턴한다. 번들 B는 리턴받은 원격 서비스 레퍼런스를 이용하여 원격 서비스 Hello를 호출하게 되고, 원격 서비스 요청을 프록시 서비스를 통해 Dispatcher 서비스에 전달된다. Dispatcher는 해당 서비스 Hello를 호출하여 실행하고 그 실행 결과를 다시 디바이스 B의 프록시 서비스에 리턴해 준다. 프록시 서비스는 번들 B에 결과를 리턴함으로써 원격 서비스 호출을 완료하게 된다.

[그림 3]은 원격 서비스 호출에 대한 시퀀스 다이어그램으로 서비스 이용자인 클라이언트 측에서 원격 서비스를 호출하였을 시, 원격 서비스가 수행되기까지의 과정을 나타낸 것이다. ServiceReference를 획득하기 위해 클라이언트 측의 RemoteServiceReference 서비스와 실제 서비스를 호출하기 위한 서비스 제공자 측의 Dispatcher 서비스를 거쳐 클라이언트에서 결과를 받는 것을 확인할 수 있다.

3. 원격 서비스 속성 정의

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 프레임워크 내에서 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. OSGi 에서는 서비스의 속성을 '속성이름=값'으로 정의한다. 원격 서비스임을 알리기 위해 'service.remote=true' 라는 새로운 속성이름과 속성값을 정의하였다.

Ⅳ. 성능평가

본 논문에서 제안한 원격 OSGi 서비스 상호운용 시스템인 RSP의 성능을 평가하고, R-OSGi와의 성능을 비교분석하였다.

R-OSGi는 분산 OSGi 프레임워크들의 서비스를 상호운용하기 위해 개발자에게 RMI 매커니즘을 제공하며 중앙 집중적인 서비스 레지스트리를 사용하는 미들웨어이다. R-OSGi를 비교 대상으로 삼은 것은, DPWS 또는 JXTA, Web Service와 같은 미들웨어 기술과 비교했을 때, 서비스 변환과정이 필요하지 않아 빠르며 컴퓨팅 자원의 소모가 적은 기술이기 때문이다.

성능평가를 위한 환경으로는 Intel Core2 (1.86GHz) CPU, 2GB RAM 이 장착된 PC 1대를 서비스 사용자 머쉰으로 사용하고, Intel Core2 Quad (2.33GHz) CPU, 3G RAM이 장착된 PC 1대를 서버 제공자 머쉰으로 사용하였다. 각각의 PC에는 JDK 1.6과 OSGi 프레임워크인 Equinox를 설치하였다.

테스트를 위해 사용된 서비스는 EchoService 이다. EchoService는 원격 서비스 사용자 측에서 문자열을 전송하게 되면 원격 서비스 제공자 측에서 문자열을 받은 뒤 똑같은 문자열을 되돌려 주는 서비스이다. 테스트 시 문자열의 길이를 Null, 512바이트, 1024바이트 일 때 각각 측정함으로써 성능을 비교하였다.

실험 결과 [표 1]과 같이 RSP는 평균 약 51.9ms, R-OSGi는 약 115.8ms의 시간이 걸려 RSP가 R-OSGi보다 약 2배 빠른 성능이 나타나는 것을 볼 수 있다.

Ⅳ. 결 론

분산 OSGi 프레임워크에서 원격 서비스의 상호운용을 가능하도록 분산컴퓨팅에서 대표적인 프로세스간 통신을 위한 미들웨어 기술인 RMI 패러다임을 적용한 원격 OSGi 서비스 상호운용방법인 RSP를 제안하였다. RSP는 P2P 방식의 서비스 발견 메커니즘을 사용하여 중앙집중적인 서비스 레지스트리 방식의 문제점을 해결하였으며, OSGi 표준 기술을 확장하여 설계되었으므로 서비스 변환과 같은 부가적인 작업이 불필요하여 JXTA나 Web Services와 같은 기존 분산 미들웨어를 활용하는 기술들과 비교했을 때 컴퓨팅 자원의 소모가 적다는 특징이 있다.

[표 1] 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

현재 OSGi는 통신, 자동차, 공장, 사무실, 가정 등 그 적용 도메인이 점차 다양해지며 보편화되는 추세이다. 그러므로 원격 OSGi 프레임워크의 서비스 공유 시 보안 뿐 아니라 서로 다른 도메인간의 상호 운용성을 지원하는 분산 OSGi 기술의 연구가 추가적으로 필요하다.

참고문헌

[1] Vijackumar 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 Allicance, “About the OSGi Service Platform”, June 2007.

[3] O. Dohndorf, 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 Compuing and Communication Workshops, 2010.

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

[5] JXTA, http://jxta.dev.java.net

[6] Web Services, http://www.w3.org/standards/webofservices/, W3C

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