OSGi 기반의 확장형 소프트웨어 플랫폼

주현태 김정국 김현주 서한석
한국외국어대학교 컴퓨터공학과
cisc@hufs.ac.kr, jgkim@hufs.ac.kr, kimhj@hufs.ac.kr, hanseok@hufs.ac.kr

Extensible software platform based on OSGi

Hyuntae Ju Jung-guk Kim Hyunjoo Kim Hanseok Seo
Department of Computer Science and Engineering, Hankuk University of Foreign Studies

요 약

기존의 비호환성 서비스 플랫폼을 대체할 수 있는 OSGi 기반의 확장형 소프트웨어 플랫폼을 제시한다. 본 논문에서는 OSGi Alliance에서 제정한 서비스 플랫폼 명세(Service platform specification)를 기반으로 플랫폼에 의존하지 않는 서비스 환경을 구현하고 이를 활용하여 홈 네트워크 환경을 위한 새로운 소프트웨어 플랫폼 구축에 관하여 논한다. 이는 플러그인 방식의 쉬운 확장 기능을 제공함으로써 서비스 제공자와 소비자의 다양한 시스템 환경에서 비용을 절감할 수 있는 효율적인 서비스 기반을 제공하게 될 것이다.

1. 서 론

정보 가전의 발달과 초고속 통신 인프라의 확충에 힘입어 생활 환경의 질적 향상을 위한 홈 네트워크가 보편화 되어 가고 있다.

홈 네트워크는 일반적으로 가정 내의 네트워크가 인터넷을 통하여 외부와 연결되는 형태를 취한다. 가정 내에서는 각 정보 기기들이 WiFi, Bluetooth, Ethernet과 같은 다양한 방법으로 연결되어 서로 커뮤니케이션이 가능하며, 이런 형태의 네트워크가 인터넷을 통해 가정 외부와도 연결이 가능한 형태를 이루고 있다. 가정 내에서 홈 네트워크를 구성하는 정보 기기에는 홈 게이트웨이와 서버, 그리고 디지털 TV, 냉/난방 및 기타 정보 가전제품과 사용자 단말기 등이 있는데, 이들은 각각 생산자에 따라 다양한 하드웨어 및 소프트웨어 환경을 가지고 있다.

정보 가전에 대한 소비자들의 다양한 요구가 생겨나고 있으며, 기기의 성능 및 기능도 다양해지고 있다. 최근 이처럼 다양한 환경과 요구 조건을 충족하기 위한 정보 가전 기기들이 개발되면서 소프트웨어의 종류와 구조가 복잡해지고 있다. 이로 인해 소프트웨어 개발이나 유지 및 보수에 소모되는 비용이 증가하여 왔고, 이는 전체 생산 비용에서 큰 비중을 차지하게 되었다. 한편, 사용자는 각 기기들마다 다양한 환경과 소프트웨어 관리 방법(소프트웨어의 활용이나 기능 확장)에 따른 불편을 감수해야 하는 문제도 있다.

본 논문에서는 이러한 문제점들을 해결하고 보다 간편하고 효율적인 소프트웨어 플랫폼을 구현하였다.

다만, 이와 같은 네트워크 환경에서 빼 놓을 수 없는 문제 중의 하나는 바로 보안인데, 이는 주제에서 벗어나므로 본 논문에서는 보안과 관련된 문제는 발생하지 않거나 이미 해결되어 있는 것으로 가정하기로 한다.

2. 서비스 환경과 소프트웨어 플랫폼의 개선

서론에서 제기한 문제들을 해결하기 위해서는 먼저 하드웨어나 운영체제와 같은 플랫폼으로부터 독립된 소프트웨어 환경이 필요하다. 사용자가 플랫폼에 구애받지 않고 자신이 원하는 기능에 따라서만 소프트웨어를 선택하여 사용할 수 있어야 하는 것이다. 그리고 기능 확장이 필요할 때 전문가가 아닌 사람도 쉽게 정보 가전 기기에 소프트웨어를 설치하여 기능을 확장할 수 있어야 한다. 필요에 따라서 사용자나 서비스 업체가 원격지에서도 가정 내의 기기에 접속하여 소프트웨어를 관리할 수 있는 환경도 제공되어야 할 것이다.

궁극적으로는 이러한 확장성과 편의성을 통해 개발자와 서비스 제공자 입장에서는 개발 및 서비스 제공에 따르는 비용을 절감할 수 있고, 사용자 입장에서는 보다 편리하게 홈 네트워크 서비스를 이용하고 관리할 수 있도록 하는 것이 목적이다.

본 논문에서는 OSGi Alliance에서 제정한 OSGi 서비스 플랫폼 명세(OSGi Sevice Platform Specification)를 활용하여 홈 네트워크 서비스를 위한

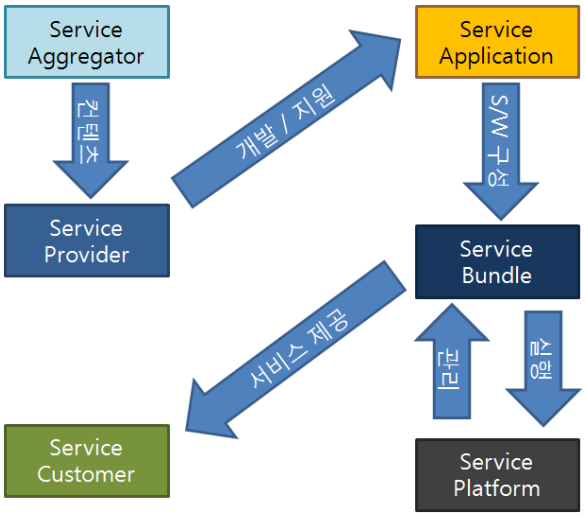
개선된 소프트웨어 플랫폼을 구현하였다.[1]

기존의 OSGi 서비스 플랫폼 위에 GUI 환경을 구축하고, 기본 GUI 틀을 동적으로 확장할 수 있는 플러그인 형태의 번들 컴포넌트를 제공한다. 또한, 이를 원격에서 제어할 수 있는 방법도 제공함으로써 기존 OSGi 플랫폼 고유의 특성과 동시에 사용자 편의성과 플랫폼의 유연성을 고려하였다.

3. OSGi 서비스 플랫폼

OSGi Alliance는 네트워크 서비스 환경에 대한 개방형 설계를 목적으로 1999년에 설립되었으며, 가정이나 자동차, 이동통신, 그리고 사무 환경이나 기타 여러 네트워크 환경에 대한 차세대 인터넷 표준화 사업을 진행 중이다. [1]

OSGi 서비스 플랫폼을 기반으로 하는 서비스의 구조와 흐름은 [그림 1]과 같이 요약할 수 있다. [2]



[그림 1] OSGi 서비스 흐름

Service Aggregator(서비스 보유자)는 제공할 서비스의 컨텐츠 자체를 보유하고 있으며, Service Provider(서비스 제공자)는 이 서비스 컨텐츠를 적당한 형태로 가공하여 네트워크를 통해 제공할 수 있도록 개발 및 지원을 하게 된다. 경우에 따라 이 둘은 통합될 수도 있다. 이 때, 서비스 보유자가 보유하고 있는 서비스 컨텐츠가 어떤 방법이나 경로로 입수된 것인지는 중요하지 않다. 자체적으로 개발 보유하는 것이어도 좋고, 다른 서비스 제공자로부터 수집한 것이어도 된다. 서비스 제공자가 개발한 서비스 프로그램은 번들(Bundle)이라고 하는데, 서비스 플랫폼(Service platform)은 이 번들의 설치와 수행에 관한 관리를 하게 된다.

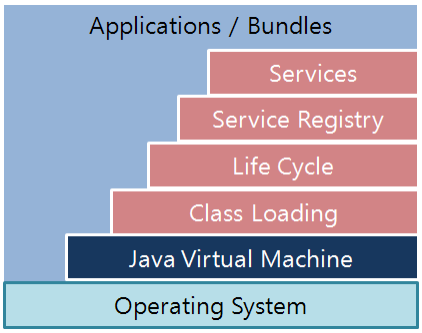
3.1. OSGi 서비스 플랫폼의 구조 및 특징

OSGi 서비스 플랫폼은 Java 기반의 소프트웨어 수행 기반으로서, OSGi Alliance에서 제정한 서비스 플랫폼 명세를 만족할 수 있도록 설계된 환경이다. 즉, 서비스 제공자와 개발자, 게이트웨이, 그리고 서비스 사용자가 OSGi라고 하는 공통적인 환경에서 서비스를 제공 및 이용할 수 있도록 해 주는 플랫폼이다. OSGi 서비스 플랫폼은 OSGi 프레임워크를 그 핵심으로 하는데, 이동통신, 텔레메틱스, 게이트웨이, 산업용 컴퓨터, 데스크탑 PC, 그리고 내장형 기기와 같은 여러 환경에서 플랫폼에 의존하지 않고 동작한다.

OSGi 서비스 플랫폼은 그 기반에서 동작하게 되는 소프트웨어 컴포넌트들을 관리하는 역할을 한다. OSGi 응용 소프트웨어인 번들은 JAR 형태로 압축되어 있는데 이것을 다운로드, 설치 및 실행할 수 있도록 하며, 필요시 업데이트를 할 수 있도록 해 준다. 물론, 모두 동일한 규격을 가진 소프트웨어들이고 프레임워크에 의해 자동으로 수행되는 기능들이므로 그 방법에 신경 쓸 필요는 전혀 없다. 또한 이러한 관리 기능을 원격에서도 수행할 수 있도록 하고 있는데, 이는 모바일과 같은 환경에서 OSGi 서비스 플랫폼을 보다 유연하게 다룰 수 있도록 해 준다.

Java 기반의 소프트웨어 구동 환경인 OSGi 서비스 플랫폼은 일반적인 Java 응용 프로그램과는 달리, 단 하나의 JVM 위에서 프레임워크와 여러 번들들이 동작하는 구조를 가지고 있다. 이러한 특성으로 인해 여러 컴포넌트(번들)들은 상호 협력 구조를 가지고 동작할 수 있으며, 심지어는 다른 업체의 컴포넌트와도 의존 관계 혹은 협력 관계를 가지고 동작할 수 있다.

각 번들은 자신이 가진 서비스 기능을 서비스 레지스트리(Service Registry)에 등록하여 여기에 등록된 서비스를 다른 번들이 이용할 수 있도록 하는 방식의 상호 협력 구조를 가진다. OSGi 서비스 플랫폼의 계층 구조는 [그림 2]와 같다.



[그림 2] OSGi 서비스 플랫폼의 계층 구조

3.2. OSGi 프레임워크와 번들

[1] 본 연구는 건설교통부 첨단도시기술개발사업 – 지능형국토정보기술혁신 사업과제의 연구비 지원(07KLSGC05)에 의해 수행되었음.

프레임워크나 번들과 같은 OSGi 소프트웨어는 Java를 기반으로 하는데, Java 환경에서는 OSGi에서 요구하는 보안이나 신뢰성, 이식성, 그리고 개방성과 같은 특성들을 대부분 제공받을 수 있기 때문이다. 한편, 마이크로소프트사의 .NET이 Java와 비슷한 특징을 가지고 있기 때문에 OSGi의 기반 환경의 후보로 거론되기도 한다. 그러나 상업용 소프트웨어라는 특성상 공개 배포 및 보편화가 불가능한데, 이는 OSGi Alliance의 의도와는 다르다. 따라서 현재로서는 OSGi의 기반 환경으로 Java 이외의 다른 대안은 없는 상황이다.

Java에서 모든 소프트웨어 컴포넌트들은 클래스 형태로 존재하는데, JVM이 클래스를 메모리에 적재하고 수행할 때는 JVM 내부의 클래스 로딩 시스템(Class loading system)이 동작하게 된다.[4] 이 클래스 로딩 시스템도 오브젝트 형태로 제공되기 때문에 이에 대한 레퍼런스(오브젝트의 주소)를 이용하여 클래스 로딩 시스템에 접근할 수 있다. 프레임워크가 직접 소프트웨어의 설치, 메모리 적재, 수행 등의 작업을 관리하므로 여러 개의 JVM이 필요하지 않고 단 하나의 JVM 위에서 프레임워크와 모든 번들들이 동작하는 형태가 된다. 이러한 구조는 위의 [그림 2]에 명시되어 있는데, [그림 2]의 'Life Cycle' 계층이 번들의 설치, 수행, 제거와 같은 작업을 관리하는 계층이다.

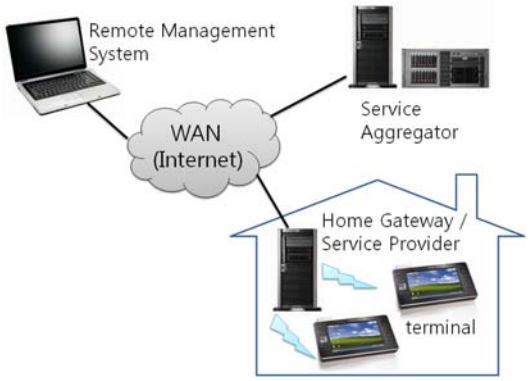
OSGi 프레임워크에서 수행되는 소프트웨어를 번들이라고 하는데, 이 번들은 OSGi 서비스 플랫폼 명세에 정의된 Bundle 인터페이스(interface)를 구현하여 제작된 소프트웨어이다. 인터페이스는 일정한 공통 양식을 기반으로 하는 클래스를 제작할 때 사용하는 추상 클래스(abstract class)의 일종인데, 프레임워크는 Bundle 인터페이스에 대한 레퍼런스를 가지고 번들에 대한 각종 관리 작업을 수행하게 된다.

4. 확장형 플랫폼

테스트를 위한 환경은 [그림 3]과 같이 구성하였으며, 사용자와 정보가전 기기(terminal)간의 대화를 위한 입출력 장치로는 LCD 터치 스크린을 이용하도록 설계하였다.

가정 내의 정보 가전 기기들은 서로 연결되어 로컬 네트워크를 구성하며, 로컬 네트워크는 인터넷과 같은 광역 네트워크를 통해 외부와 연결된다. 가정 내의 로컬 네트워크는 외부와 가정 내의 기기들을 연결해 주는 홈 게이트웨이와 이 게이트웨이를 통해 외부 혹은 가정 내의 다른 기기들과 서비스를 주고 받게 될 사용자 기기(단말기)로 구성된다. 홈 게이트웨이는 사용자와 서비스 제공자 사이의 접속을 중재하는 역할을 하며, 기기를 외부에서 통제할 수 있도록 원격 접속 시스템과 중재해 주는 역할도 한다. 단, 서론에서 밝힌 것과 같이 이러한 네트워크 환경에서 반드시 발생하게 되는 보안과 관련된 문제는 본 논문의 주제를 벗어나므로 무시하기로 한다.

한편, 서비스 제공자(Service Provider)는 사용자가 원하는 서비스 컨텐츠를 가진 적절한 서비스 보유자(Service Aggregator)를 찾아 해당 서비스 컨텐츠를 얻어 오게 된다. 그리고 그 것을 사용자에게 제공하기로 약속한 양식(데이터 구조)에 맞게 가공하여 사용자에게 서비스를 제공한다. 본 논문의 테스트 환경에서는 시스템 구성의 간소화를 위해 홈 게이트웨이와 서비스 제공자의 기능을 통합하기로 하였다.



[그림 3] 테스트 환경

서비스에는 여러 가지 형태가 있으나 본 논문에서는 날씨 정보 서비스를 가지고 예를 들어 소개하기로 한다. 예제로 제작한 날씨 서비스에서 서비스 보유자는 Weather.com 사이트이며, RSS로 각 지역별 날씨를 제공하고 있다. RSS로 제공되는 날씨 정보는 XML 형태로 되어 있는데, 서비스 제공자는 이 XML 형태의 정보를 분석하여 서비스 번들에 정의되어 있는 양식으로 제공한다.

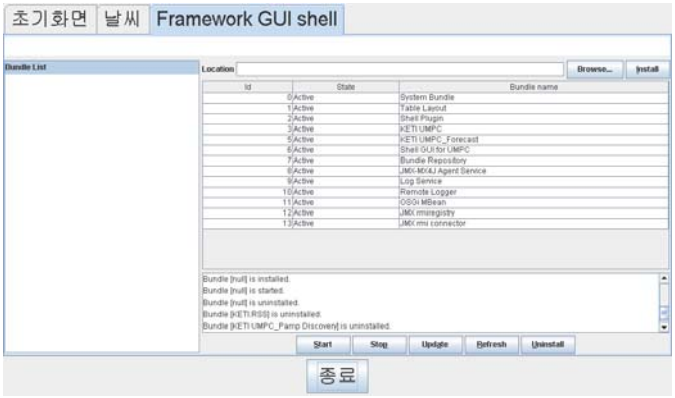
4.1. 사용자 인터페이스(User Interface)

기본적인 사용자 인터페이스(User Interface)는 [그림 4]와 같다. 초기 화면으로는 선택한 지역의 간단한 날씨 정보와 현재 시각을 보여 준다.

현재 사용 가능한 서비스는 화면 상단의 탭으로 보여주며, 이 탭을 선택하면 해당 서비스 화면으로 이동할 수 있다.

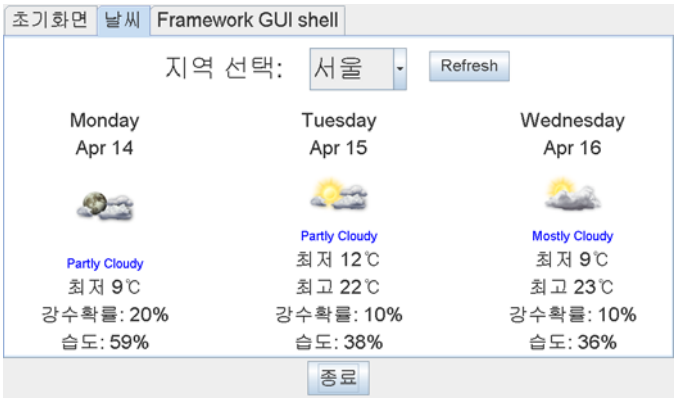
[그림 4] 사용자 인터페이스

기본적으로 초기 화면과 프레임워크를 관리할 수 있는 Framework GUI shell을 제공한다. GUI shell은 현재 설치되어 있는 번들의 목록과 상태, 번들을 찾아서 설치할 수 있는 탐색 창, 그리고 프레임워크의 작업 내역을 표시하는 화면을 제공한다. 이 GUI shell을 이용하여 번들의 설치, 수행, 업데이트 및 제거 작업을 수행할 수 있다. 번들은 로컬 디스크는 물론이고 웹 서버나 FTP와 같이 네트워크를 통해 접근할 수 있는 장소에 있으면 별도의 다운로드 과정 없이 로컬 디스크에 존재하는 것처럼 곧바로 설치가 가능하다.



[그림 5] Framework GUI shell

서비스 이용의 한 가지 예로서 날씨 정보를 제공하는 번들을 설치했을 때, 번들을 설치하면 상단 탭에 날씨 서비스 화면으로 이동할 수 있는 탭이 생성된다. 아무것도 설치되지 않은 [그림 4]에서는 기본 탭 항목만 보이지만, [그림 5]에서 GUI shell을 이용하여 날씨 서비스 번들을 설치 및 시작한 후에 탭의 변화를 발견할 수 있을 것이다. 새로 추가된 탭을 선택하면 [그림 6]과 같이 날씨 서비스를 이용할 수 있는 화면을 볼 수 있다.



[그림 6] 서비스 예제 - 날씨 서비스

4.2. 플러그인(plug-in) 컴포넌트

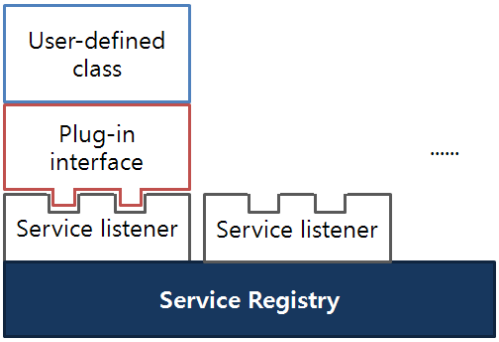
OSGi 프레임워크에서 번들은 플러그인 형태로 동작한다. JAR 형태로 묶여 있는 번들 파일만 프레임워크에 등록하면 프레임워크의 기능이 확장되는 형태로 새로운 서비스 기능이 추가되는 것이다. 이는 전혀 새로운 소프트웨어를 설치하는 과정에 비해 훨씬 간편하다.

본 논문에서 제시하는 확장 플랫폼과 플러그인의 동작 방법은 OSGi 프레임워크와 번들의 동작 방법을 그대로 상속하였다. 플러그인 역시 하나의 OSGi 프레임워크 기반의 번들이면서 동시에 확장 플랫폼에서 정의한 인터페이스를 가지고 있는 컴포넌트이다. 이 인터페이스는 getName()과 getGUI() 메소드를 제공하는데, 이는 각각 화면 상단 탭에 표시할 플러그인의 이름과 화면 가운데 표시할 GUI 컨테이너(JPanel)를 리턴하는 메소드이다.

확장 플랫폼은 플러그인 인터페이스에 대한 서비스 리스너(Service Listener)를 정의하고 있다. 이 서비스 리스너는 일종의 이벤트 핸들러인데, 프레임워크에 번들이 설치되면 그것이 특정 인터페이스를 가진 것인지를 감지하는 일을 한다. 이해하기 쉽게 표현하면 Service Listener는 콘센트, 플러그인 인터페이스는 플러그와 같다고 볼 수 있다. 콘센트에는 그에 맞는 플러그만 꽂을 수 있는 것과 같다.

서비스 리스너는 인식한 플러그인을 서비스 레지스트리(Service Registry)에 등록한다. 더 정확히 말하면, Service Registry에는 등록하고자 하는 플러그인의 이름과 인터페이스가 저장된다. 나중에 이 서비스 레지스트리에서 특정 플러그인을 찾을 때는 플러그인의 이름이나 인터페이스를 키(key)로 하여 찾아낼 수 있는데, 주로 먼저 인터페이스로 찾은 것들 중에서 이름을 가지고 다시 찾아 내는 방법을 사용한다.

플러그인의 구조와 Service Listener, 그리고 Service Registry의 관계를 [그림 7]에 간단히 나타내었다.

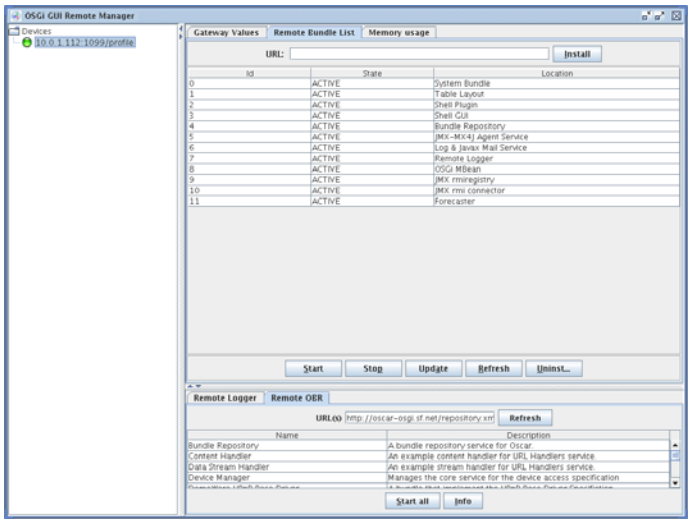


[그림 7] 플러그인 구성

서비스 리스너는 서비스 레지스트리에 플러그인을 등록하는 동시에 getGUI() 메소드에서 얻은 이름으로 탭을 생성한다. 탭이 선택되면 서비스 레지스트리에서 그 탭에 해당하는 플러그인을 찾아 getGUI() 메소드로 얻은 GUI 컨테이너의 내용을 화면에 보여준다.

4.3. 원격 관리 시스템

본 논문에서 제시한 확장 플랫폼에 설치되는 플러그인과 기타 번들들은 직접 해당 기기를 이용하여 제어할 수도 있지만, 외부에서 원격 관리를 할 수 있는 방법도 제공하고 있다. 이 때는 원격 관리 클라이언트가 필요하며, 이 클라이언트는 홈 게이트웨이로부터 기기들의 목록을 얻어 온다. 그리고 게이트웨이와 기기들의 목록(IP로 표시)을 트리 형태로 보여 주고 사용자가 각 기기를 선택하여 관리할 수 있도록 해 준다. 이렇게 구축된 원격 관리 시스템을 이용하여 원격에서 사용자의 기기에 플러그인이나 번들을 설치, 수행 및 제거하는 등의 작업을 할 수 있고, 선택한 기기의 메모리 사용률 등을 확인할 수도 있다.



[그림 8] 원격 관리 화면

한편, 원격 관리를 받게 되는 각 기기들은 서버 역할을 하는데, 원격 관리 클라이언트로부터 오는 작업 요청을 수행한다. 각 기기는 프레임워크가 수행되고 있는 JVM의 클래스 로딩 시스템과 프레임워크에 설치된 번들 정보와 같은 각종 정보들을 모아서 원격 클라이언트로 전송한다. 원격 클라이언트는 각 기기들의 이러한 정보를 가지고 원격 관리 작업을 수행하게 된다. 클래스 로딩 시스템 오브젝트는 네트워크를 통해 다른 곳으로 전송할 수도 있다. 그렇기 때문에 원격의 클래스 로딩 시스템을 이용하여 그 곳의 JVM을 다룰 수 있다.

5. 결론

여러 서비스 제공자, 그리고 다양한 환경을 가진 기기들이 네트워크상에서 서비스를 원활하게 제공하기 위해서는 일관성 있는 서비스 구조와 통일된 소프트웨어 플랫폼이 필요하다. 각 기기들의 환경에 맞는 소프트웨어를 각각 개발하는데 따르는 비용을 절감하기 위해서는 이식성이 좋은 플랫폼도 있어야 한다. 한편, 사용자에게는 소프트웨어의 이식성과 기기 생산자 특성에 따른 불편이 따르지 않아야 한다.

본 논문에서 제시한 OSGi 기반의 확장형 소프트웨어 플랫폼은 이러한 문제들을 해결할 수 있는 실마리를 제공할 것이다. 플랫폼에 의존하지 않는 Java 환경을 이용함으로써 이식성 문제를 해결하였고, OSGi 서비스 플랫폼을 기반으로 하여 서비스 구조의 일관성 문제도 해결하였다. 그리고 쉽게 확장할 수 있는 GUI 환경을 제공하고 서비스 번들을 플러그인 컴포넌트 형태로 다룰 수 있게 함으로써 사용자 측면의 편의성 문제도 해결하였다.

본 논문에서는 기술적인 측면에서만 접근하였지만, 추후 연구를 통해서는 보다 나은 사용자 인터페이스(UI)와 네트워크 환경에서 발생할 수 있는 여러 가지 예외 상황(보안 등)에 대한 대처 방법을 마련함으로써 실용화에 대한 준비를 해 나갈 것이다.

참고 자료

[1] OSGi Alliance, Service Platform Core Specification release 4, p. 1, 2006.
[2] OSGi Alliance, Service Platform Specification release 3, pp. 15-21, 2003.
[3] OSGi Alliance, Technical White Paper, p. 11, 2007.
[4] Jon Meyer, Troy Downing, “JAVA Virtual Machine”, O’REILLY, pp. 63-68, 1997.