논문 2005-42CI-5-5

OSGi 기반 이동 에이전트 관리 시스템 설계

(Design of Mobile Agent Management System based on OSGi)

이 승 근*, 김 인 태*, 김 태 간*, 이 경 모*, 임 기 욱**, 이 정 현***

(Seungkeun Lee, Intae Kim, Taegan Kim, Kyoungmo Lee, Kiwook Rim, and Junghyun Lee)

요 약

OSGi 프레임워크는 다양한 가전, 센서, 디바이스 등의 상호 운용을 가능하게 함으로써 지능형 홈네트워크의 기반 기술로 주목 받고 있다. OSGi 프레임워크에서는 서비스의 배포 및 설치 단위인 번들 코드의 다운로드를 통한 원격 설치를 지원하지만, 실행 중인 번들의 이동을 지원하지 못한다. 따라서 OSGi 프레임워크 상에서 실행 중인 번들을 다른 프레임워크 상으로 이동할 수 있는 이동 에이전트 형태의 번들을 지원해야 하며, 이를 위한 별도의 시스템이 필요하다. 본 논문은 OSGi 프레임워크들간 개체의 이동성을 보장하기 위해 동적 에이전트 형태의 번들 관리 방법을 제안한다. 이를 위해서 에이전트의 라이프 사이클과 이동 관리 기능을 위한 이동 에이전트 관리 시스템을 설계한다. 설계한 이동 에이전트 관리 시스템은 OSGi 프레임워크에서 동작될 수 있기 위해서 번들 형태로 구현되며, 자율적인 서비스에 대한 동적 관리를 가능하게 함으로써 편재형 환경에서의 이동성을 보다 효과적으로 지원할 수 있다.

Abstract

OSGi(Open Service Gateway Initiative) offers a unique opportunity for pervasive computing as a potential framework for achieving interoperability between various sensors, home appliances, and networked devices. The OSGi framework supports a remote installation of a bundle, which is a unit that installs and deploys services. However, in order for the bundle in execution to migrate, a specific form of bundle such a mobile agent, is needed one which is able to move through a heterogeneous network. This paper proposes a method that can manage bundles like these dynamic agents, in order to ensure the mobility of entities in multiple OSGi framework environments. For our purposes, we have designed the mobile agent management system for managing the lifecycle and mobility of agents in the OSGi framework. The mobile agent management system we are proposing implements a bundle form which can perform in an OSGi framework as well as manage autonomous mobile services. As a result, mobility in a pervasive environment will be supported more efficiently.

Keywords : OSGi, 이동 에이전트, 홈 네트워크

I. 서 론

편재적인 컴퓨팅 환경을 위해서 실세계의 센서와 가전제품들을 상호 연결할 수 있는 DLNA, UPnP, Jini, HAVi, OSGi 등의 개방형 표준 인터페이스에 대한 많은 연구들이 진행되고 있는데, 이러한 최근의 동향은 유비쿼터스 컴퓨팅에 대한 실질적인 편재형 컴퓨팅 개념과 같은 맥락에 놓여 있다[1]. 특히, OSGi는 센서, 임베디드 컴퓨팅 기기, 가전제품들을 인터넷에 접속하는 표준방식에 관한 산업계의 계획으로 서비스 제공업체와 가정 및 소규모 기업망 내에 있는 장치들 사이에서 통신 및 제어를 할 수 있게 하기 위한 자바 기반의 개방형 표준 프로그래밍 인터페이스를 제공한다[2]. 이전의 이러한 기술들이 디바이스들 간의 상호 운용성에 집중한 반면에, OSGi는 디바이스에 관한 서비스의 전달, 배치, 관리 등에 초점을 맞춘다. 또한 OSGi 기반의 어플

* 학생회원, *** 평생회원, 인하대하교 컴퓨터정보공학부
(Dept. of Computer Sci. & Information Eng., Inha Univ.)
** 평생회원, 선문대학교 컴퓨터정보학부
(Dept. of Computer & Information Sci., Sunmoon Univ.)
※ 본 연구는 정보통신부 대학 IT 연구센터 육성 지원 사업의 연구 결과로 수행되었습니다.
접수일자: 2005년8월17일, 수정완료일: 2005년9월6일

리케이션 개발을 통한 Jini나 UPnP 등의 기술에 관한 연결 서비스를 배치할 수 있거나 상호 작용이 가능하다. 이미 OSGi는 홈, 오피스, 자동차와 같은 로컬 네트워크 상에서 다양한 디바이스들을 연결하며, 관리 가능한 확장성 있는 프레임워크를 제공함으로써, 유비쿼터스 컴퓨팅 환경에 한발 앞서 다가가고 있다. 또한, 표준적인 실행 환경과 서비스 인터페이스를 정의함으로써, 서로 다른 다양한 종류의 리소스들로부터 서비스와 디바이스들의 동적인 발견과 협력을 증진시킨다[3][4].

OSGi 기반의 시스템에서는 새로운 서비스를 분배할 수 있는 구조를 가지고 있으며, 그 구조가 로컬 네트워크상의 구성 요소들만으로 이루어져 있어서 비교적 폐쇄적인 특성을 갖는다. 하지만, 그러한 하나의 OSGi 프레임워크 내에서의 서비스 관리와 분배가 동적으로 수행될 수 있는 반면에, 복수의 프레임워크 간의 이동성을 갖는 응용들에 대한 지원은 부족한 실정이다. 그래서 확장된 편재형 공간에서 복수의 OSGi 프레임워크들 간의 사용자, 디바이스 및 센서들의 이동성에 관한 충분한 고려가 이루어져야 하며, 그에 따른 이동성을 지원하는 서비스에 관한 연구가 필요하다. 편재형 공간에서 사용자, 디바이스 및 센서들은 이동성을 가지므로 자신의 실행 상태를 저장한 채 서비스가 이동할 필요성이 있다. 또한, 다차원적인 OSGi 프레임워크가 위치하는 확장된 편재형 공간 상에서 이러한 서비스 개체들의 이동에 따른 관리를 효과적으로 지원해야 한다.

네트워크상에서의 지능형 서비스를 구현하는 에이전트 기술은 독립된 에이전트들 간의 통신과 업무 협력을 통해서 이루어진다[5][6][7]. 자율적인 코드 이동성을 지원하는 이동 에이전트 기술은 새로운 서비스를 분배할 수 있는 OSGi 기반의 네트워크 시스템 구조 내에서 서비스 이동을 위한 구현을 가능하게 한다.

본 연구에서는 분산 환경에서 상태 정보를 갖고 이동 및 복제 등을 지원하는 이동 에이전트 기술을 이용한 OSGi 기반 프레임워크를 설계한다. 설계한 프레임워크는 이동 에이전트 형태의 번들을 지원함으로써 OSGi 시스템 간의 번들의 자유로운 이동을 지원한다. 따라서, 특정 컴포넌트나 사용자를 위한 서비스, 디바이스 드라이버 등 다양한 형태의 개체들의 이동성을 지원할 수 있다. 이를 위해서 OSGi 프레임워크의 오픈 소스인 Knopflerfish 1.3.3[8][9]를 확장해서, 번들의 이동 라이프 사이클을 지원하기 위한 이동 에이전트 관리 시스템을 설계한다.

Ⅱ. 관련 연구

OSGi는 네트워크 환경에서 서비스를 전달하고 배치, 관리하기 위한 표준 명세를 정의하는 비영리 단체이다. 초기에는 홈 서비스 게이트웨이에 집중되었지만 최근에는 특정 네트워크 환경에 국한하지 않고 유비쿼터스 환경까지 확장해 가고 있다. 따라서, 다양한 임베디드 디바이스와 이를 이용하는 사용자를 위한 서비스 게이트웨이 구축을 목표로 하고 있다[10].

그림 1의 OSGi 서비스 플랫폼은 OSGi 프레임워크와 표준 서비스들로 구성된다. OSGi 프레임워크는 이러한 서비스들을 위한 실행 환경을 의미하며 최소한의 컴포넌트 모델, 컴포넌트를 위한 관리 서비스, 서비스 레지스트리 등을 포함한다. OSGi 프레임워크는 번들이라고 불리는 OSGi 컴포넌트를 설치하고, 서비스 등록 및 실행을 위한 프로그래밍 모델을 지원한다.

또한 프레임워크 자신도 번들로 표현되며, 이러한 번들을 시스템 번들이라고 한다. 번들을 위한 호스팅 환경을 제공하는 OSGi 프레임워크의 역할을 정리하면 다음과 같다.

- 번들의 라이프 사이클 관리
- 번들들 간의 상호 독립성 해결
- 서비스들에 관한 레지스트리 관리
- 번들의 상태 변화, 서비스 등록 및 해제, 프레임워크의 동작 등에 관한 이벤트 처리

번들들은 서비스 레지스트리에 등록된 서비스를 이용하는 서비스 집합인 동시에 컴포넌트 단위이다. 서비스 구현은 물리적이며 논리적인 단위인 번들을 통해서 프레임워크에 전달되고 배치되어진다. 물리적으로, 번들은 코드, 리소스, Manifest 파일 등을 포함하는 Java Archive (JAR) 파일 형태로 배포된다. 특히, 번들의 manifest 파일은 프레임워크에게 번들 클래스의 실행

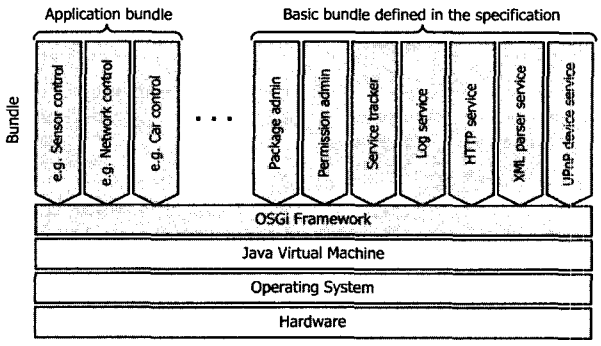


그림 1. OSGi 서비스 플랫폼
Fig. 1. The OSGi Service Platform.

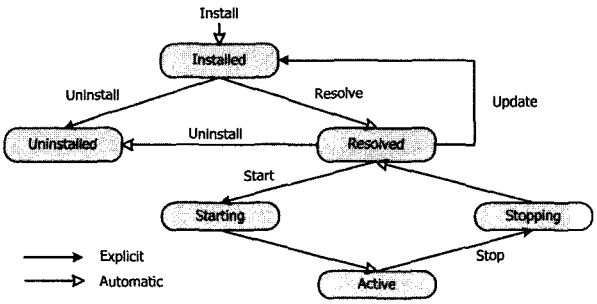


그림 2. 번들의 라이프 사이클
Fig. 2. The life cycle of the bundle.

경로를 알려주며, 다른 번들들과 공유하게 될 자바 패키지를 선언한다. 또한, 번들의 Activator 클래스에 대한 정보를 갖고 있다. 논리적으로, 하나의 번들은, 운영체제에서의 하나의 프로세스와 유사한 개념으로, 어떤 서비스를 제공하는 서비스 제공자(provider)이거나 실행 시간에 프레임워크 내에 어떤 서비스를 이용하려는 서비스 요청자(requester)이다. OSGi 프레임워크는 번들의 라이프 사이클을 관리하는 메커니즘을 제공하는데, 만약 번들이 인스톨되어 프레임워크 상에서 실행되면 자신의 서비스를 제공할 수 있게 된다. 또한 프레임워크 상에서 실행 중인 다른 서비스를 발견하여 이용할 수 있으며, 프레임워크의 서비스 레지스트리를 통해서 다른 번들들의 서비스들과 묶여질 수 있다. 번들은 OSGi 프레임워크의 서비스 레지스트리에 서비스를 등록할 때, 서비스에 관한 속성-값(attribute-value) 쌍의 형태로 그 서비스의 특성을 함께 저장하는데, 이러한 서비스 특성은 동일한 서비스를 제공하는 여러 서비스 제공자들 사이에서 구분될 수 있도록 사용된다.

JAR 파일로 배포된 번들은 인스톨 후 시작되면 Activator 클래스에 의해 동작하게 된다. 번들의 라이프 사이클은 그림 2와 같으며, 대부분의 경우 프레임워크의 관리 메커니즘에 의해 직접적인 영향을 받는다. 번들은 인스톨된 후 사용되기 전 반드시 resolved 상태로 준비되어야 하는데, 프레임워크는 외부의 자바 패키지들에 관한 의존성을 체크함으로써 번들을 resolved 상태로 만든다. 이는 만약 어떠한 의존성이 존재한다면, 프레임워크는 요구되는 번들이 사용 가능한지 또는 번들에 대해서 접근 가능한지를 확인하는 것이다. 번들은 Activator 클래스의 BundleActivator 인터페이스를 구현함으로써 개발자에 의해 구현된 특정 작업을 수행하게 된다. 프레임워크에 의해 호출되는 Activator 클래스는 start와 stop 메소드와 함께 구현된다. 이러한 메소드들은 일단 호출되면, 다른 번들에의 접근, 번들 관리 작업 수행, 서비스 등록, 다른 서비스 검색, 서로 다른 이벤트들에 대한 리스너 등록 등과 같은 프레임워크 기능들을 가능케 하는 실행 환경인 Bundle Context을 얻게 되고 번들은 이를 통해서 간접적으로 OSGi 프레임워크의 기능들에 접근할 수 있다.

OSGi 서비스는 인터페이스와 서비스 구현으로 구성되어 있다. 서비스들 사이의 접근은 인터페이스를 통해 이루어지며 각 서비스는 서비스 레지스트리에 객체 단위로 등록된다. OSGi 프레임워크 상의 서비스를 위한 어플리케이션은 번들을 개발함으로써 이루어진다. 번들 개발을 위한 프로그래밍 모델은 컴포넌트 기반이며, 프레임워크는 어플리케이션 개발자에게 프레임워크와 서비스 간의 인터페이스만을 정의함으로써 일관된 프로그래밍 모델을 제공한다. 실제 구현은 번들 개발자의 몫이며, 서비스 선택과 상호 작용에 관한 결정은 서비스 발견과 실행 시간에 동적으로 실현된다. 이러한 서비스 정의와 구현의 분리는 서로 다른 서비스 제공자들로부터 제공된 서비스들 간의 상호운용을 가능케 한다.

OSGi 프레임워크에서 제공하는 번들의 라이프 사이클 관리와 클래스 로딩에 대한 지원은 다른 번들이나 외부로부터의 코드를 이용할 수 있도록 설계된 구조이다. 이러한 특징으로 인해 번들의 생성부터 소멸까지의 상태 정보가 관리되며, 자바 클래스 파일로부터 새로운 인스턴스를 생성할 수 있는 기능을 제공할 수 있다[11][12].

III. 시스템 설계

1. OSGi 기반 이동 에이전트 관리 시스템 구조

이 장에서는 OSGi 프레임워크 상에서 이동 에이전트 번들을 관리할 수 있는 OSGi 기반 이동 에이전트 관리 시스템을 설계한다. 그림 3은 이동 에이전트를 관리하기 위해 추가된 AgentManager번들 구성요소를 나타낸다.

전체 구조는 이동성 관리를 위한 Mobility Interface, 직렬화/비직렬화를 처리하는 요소, SOAP 메시지 송/수신을 위한 요소로 구성되어 있다.

MobileBundleManager가 이동에이전트의 이동요청을 받으면 Mobility Interface 통해 에이전트의 라이프사이클을 관리한다. 이동전 상태 정보는 Serializer에 의해 XML로 직렬화되며 SOAPClient는 SOAP 메시지로 목적지로 전달한다.

클래스 파일은 목적지 BundleInstaller를 통해 설치되

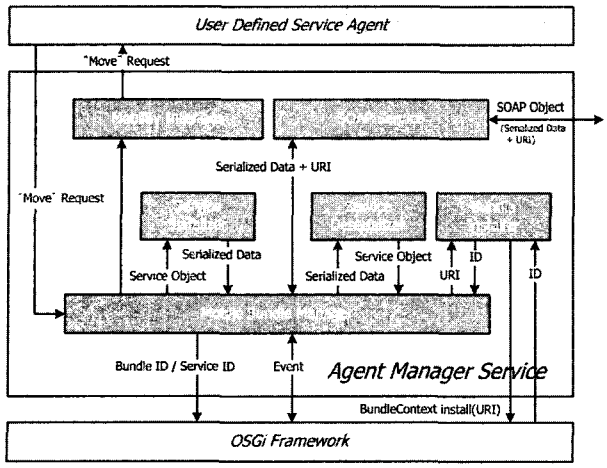


그림 3. OSGi 기반 이동 에이전트 관리 시스템 구조
Fig. 3. The structure of the mobile agent management system based on OSGi.

며 Deserializer는 SOAP메세지 정보를 객체로 비직렬화하여 서비스를 재개한다.

2. 이동 에이전트 등록 및 생성

그림 4는 이동 에이전트의 등록 및 생성 과정을 보여준다. 사용자 정의 에이전트를 위해서 OSGi의 BundleActivator를 상속받은 MobileBundleActivator를 이용한다. 사용자 정의 에이전트 번들이 활성화되면 MobileBundleActivator는 EAR(Exported Agent Repository)에 그 에이전트의 정보를 등록한다. 즉, 구현되는 에이전트 번들은 MobileBundleActivator를 자신의 BundleActivator로 지정하고, Manifest 파일에 기록된 Agent-Class, Agent-Name 헤더 값을 이용하여 EAR에 자신을 등록한다.

새로운 에이전트 생성을 요청하면 AgentManager는 EAR에서 해당 에이전트를 찾아 초기화 하고

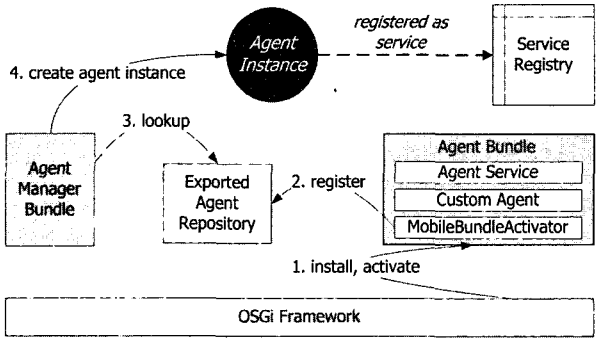


그림 4. 이동 에이전트 등록 및 생성
Fig. 4. The registeration and creation of the mobile agent.

BundleContext를 이용하여 OSGi 서비스로 등록한다.

3. 이동 에이전트 이동

이동 요청을 받으면 에이전트의 상태를 이동요청 상태로 변경한 후 실행을 멈출 것을 요청한다. 요청을 받은 에이전트는 현재 처리 중인 액션 수행을 마친 후 리턴한다.

이동 가능한 상태로 전환된 경우 Serializer를 통해 상태정보를 XML형태로 직렬화시키고, SOAPClient에게 에이전트 이동을 요청한다. SOAPClient는 이동 목적지를 확인하고 SOAP 메시지를 생성하여 목적지에 SOAPService의 service를 호출한다. 이동이 성공했다면 현재 수행 중이던 서비스를 레지스트리에서 삭제한다.

목적지에서는 SOAP 메세지를 기다리고 있던 SOAPService가 클래스가 위치한 URL정보와 직렬화된 데이터를 수신하고 이를 MobileBundleManager에게 전달한다. MobileBundleManager는 전송받은 객체를 비직렬화하기 전에 BundleInstaller를 통해 원격지로부터 번들을 설치하고 설치 성공 후 객체를 비직렬화(Deserialization)하여 이동 전 상태로 복구시킨다. 마지막으로 에이전트를 RUNNABLE 상태로 전환한 후 서비스 레지스트리에 등록한다.

4. 이동 에이전트 라이프 사이클 관리

이동 에이전트는 그림 5와 같은 라이프 사이클을 가진다.

서비스가 Service Registry에 등록되면 초기화 상태인 INIT 상태가 되며 서비스 시작과 동시에 RUNNING 상태로 전환된다. 다음 수행 할 액션이 있는 경우 RUNNING 상태를 유지 시키고, 더 이상 수행할 액션이 없는 경우 서비스 타입에 따라 두 가지 형태로 상태 전환을 하게 된다. 첫째, 일반적인 상태(TYPE=NORMAL)인 경우 더 이상 수행하지 못하는 STOP 상태로 전환되고 서비스는 Service Registry에서 등록을 해제한다.

또 다른 형태는 데몬 상태(TYPE=DAEMON)인 경우 다음에 수행할 액션이 추가될 때까지 WAIT 상태를 유지한다.

RUNNING 상태 또는 WAIT 상태에서 이동하라는 요청을 받은 경우 MOVABLE 상태로 전환된다. RUNNING 상태에서 이벤트를 받은 경우 현재 처리중인 액션을 다 처리한 후 MOVABLE 상태로 전환 되며

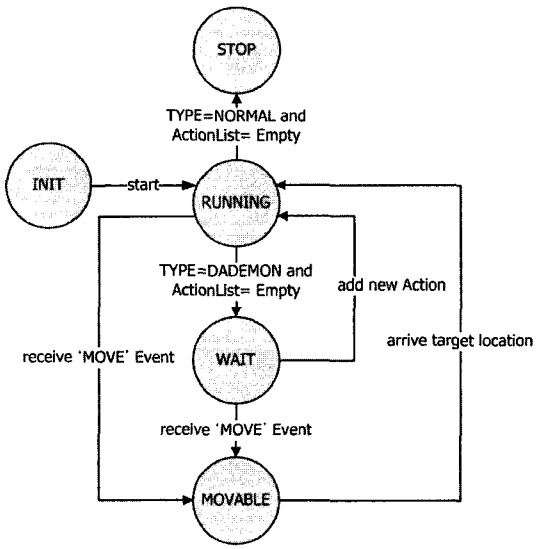


그림 5. 이동 에이전트 라이프 사이클
Fig. 5. The life cycle of the mobile agent.

WAIT 상태에서는 바로 전환 된다.

Ⅳ. 시스템 구현

본 논문에서 제안한 OSGi 기반 이동 에이전트 관리 시스템을 구현하기 위하여 사용된 전체 소프트웨어 구성도는 표 1과 같다.

Konpflerfish에서 Apache Axis를 이용하여 SOAP 전송시 자바 1.5 이상 버전에서 작은 버그가 발생되어 1.4.2 버전을 이용하여 작성하였다.

객체 이동시 그림 6과 같이 XML형태로 직렬화하여 SOAP Body 엘리먼트에 파라미터로 전달하였으며 직렬화와 SOAP전송을 위해 각각 Castor와 Apache Axis를 이용하였다. 그 밖에 리소스 파일과 클래스 파일은 SOAP 첨부 형태로 전송하지 않고 원격지의 번들 배치 기능을 이용하여 이동 에이전트 번들을 설치하였다.

본 논문에서 구현한 프레임워크를 이용하여 이동 가능한 번들을 작성하기 위한 절차 및 설정 다음과 같다. 우선 사용자 정의 에이전트 클래스를 구현하기 위해서는

표 1. 소프트웨어 구성도
Table 1. Software configuration.

SOAP	Apache Axis 1.2
Java Binding	Castor-0.97
OSGi Framework	Knopflerfish 1.3.3
Java Virtual Machine	Java 1.4.2
Operating System	Windows XP

```
<soapenv:Envelope ....>
  <soapenv:Body>
    <ns1:service xmlns:ns1="http://webserivce.ema">
      <obj xsi:type="xsd:string">
        &lt;agent&gt;
        &lt;status&gt;5&lt;/status&gt;
        &lt;action&gt;
        &lt;name&gt;Simple Action&lt;/name&gt;
        &lt;msg&gt;goodluck&lt;/msg&gt;
        &lt;/action&gt;
        &lt;/agent&gt;</obj>
    </ns1:service>
  </soapenv:Body>
</soapenv:Envelope>
```

그림 6. 전달 SOAP 메시지 형태 예
Fig. 6. The example of Transferring SOAP Message.

추후 Castor를 이용한 객체 직렬화를 위하여 XML 스키마를 작성하여야 하며 이는 에이전트 번들에 정의된 Agent 클래스를 상속받는 형태로 작성되어야 한다.

사용자 정의 에이전트의 Manifest 파일은 Bundle-Activator를 MobileBundleActivator로 지정하였으며 이를 위해 Agent 번들에 있는 ema.core.activator 패키지를 임포트하였다. MobileBundleActivator 클래스는 Bundle 객체를 이용하여 Manifest에 기록되어 있는 Agent-Class, Agent-Name 헤더 값을 읽고 이를 EAR에 등록한다. AgentManager 번들에서 사용자 에이전트를 서비스로 등록하기 위해서는 EAR에 등록된 에이전트 클래스를 동적으로 로드하여야 한다. 이를 위해 사용자 정의 에이전트는 agent.impl 패키지로 한정 지었다. 다음은 Manifest 파일의 예이다.

```
Bundle-Activator:ema.core.activator.
    MobileBundleActivator
Agent-Name: MyAgent
Agent-Class: agent.impl..MyAgent
Import-Package: org.osgi.framework,
    ema.core.activator
...
```

Ⅴ. 실 험

본 논문에서 제안한 서비스 이동 프레임워크를 실험하기 위해서 MP3 음악을 재생하는 MPlayer 번들을 개발하였다. 그림 7과 같은 실험 환경으로 PDA와 PC에 각각 JVM, OSGi, 이동에이전트프레임워크 번들을 설치하였다.

PDA에서 MPlayer를 이용하여 음악을 듣고 있던 사

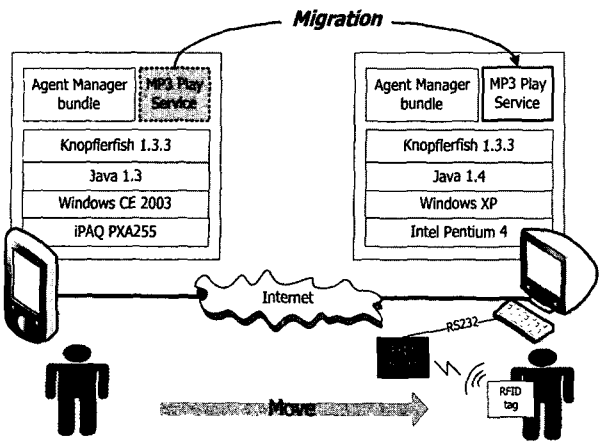
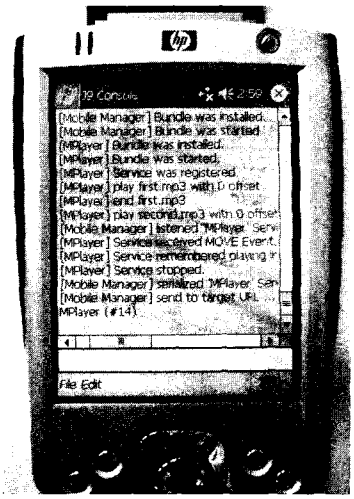


그림 7. MPlayer 번들의 이동
Fig. 7. The mobility of the MPlayer bundle.

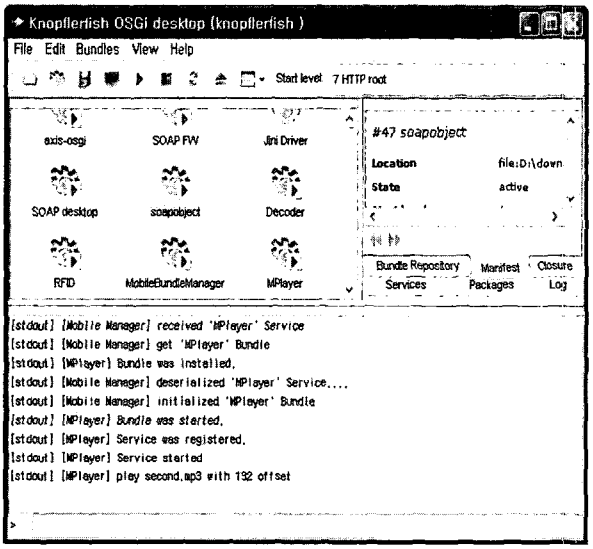
용자가 PC가 있는 공간으로 이동하였을 경우 MPlayer 서비스를 PC로 이동하고, PDA에서 듣던 음악의 위치부터 다시 시작하여 사용자에게 지속적인 MPlayer 서비스를 제공하였다.

그림 8(a)는 서비스 이동전 PDA에서 MPlayer 서비스를 이용하는 화면이다. 이동시 직렬화하는 상태 정보는 음악 목록과 현재 재생하는 곡의 offset 정보이며, MPlayer는 GUI를 가지고 있지 않고 간단한 환경설정 파일을 통해 mp3 목록을 재생하는 번들이다.

그림 8(b)는 번들 이동 후의 결과 화면을 보이고 있다. 번들의 클래스 로딩 기능을 이용하여 MPlayer 번들은 자동으로 다운로드 받아 설치되고, 음악 목록과 offset 정보로 서비스를 초기화하고 음악을 재생한다.



(a) 이동 에이전트 번들의 이동 전 화면
(a) Before moving of Mobile agent bundle



(b) 이동 에이전트 번들의 이동 후 화면
(b) After moving of Mobile agent bundle

그림 8. MPlayer 번들의 이동 화면
Fig. 8. The shots of the MPlayer bundle's mobility.

VI. 결 론

홈 네트워크 등 편재적인 컴퓨팅 환경을 구성하는 OSGi 프레임워크들간 개체의 이동을 보장하기 위해서는 에이전트 기술의 적용은 불가피하다. 본 논문에서는 OSGi 프레임워크 상에서 자율적으로 실행될 수 있는 이동 에이전트 형태의 번들을 제안하였고, 이를 위해서 이동 에이전트의 라이프 사이클과 이동 관리 기능을 위한 이동 에이전트 관리 시스템을 설계하고 구현하였다. 설계한 이동 에이전트 관리 시스템은 번들 형태로 구현되어 OSGi 프레임워크에서 동작될 수 있었으며, 자율적인 서비스에 대한 동적 관리를 가능하게 함으로써 이동성을 보다 효과적으로 지원할 수 있었다.

향후 지능형 서비스를 제공하기 위해서 이동 에이전트 기술을 이용한 OSGi 기반의 상황 인식 프레임워크에 대한 연구와 이동 에이전트의 보안에 대한 고려가 이루어져야 할 것이다.

참 고 문 헌

[1] Open Services Gateway Initiative: http://www.osgi.org.
[2] D. Marples and P. Kriens, "The Open Services Gateway Initiative: An Introductory Overview," IEEE Communications Magazine, Vol. 39, No. 12, pp. 110-114, December 2001.
[3] C. Lee, D. Nordstedt, and S. Helal, "Enabling Smart Spaces with OSGi," IEEE Pervasive Computing, Vol. 2, Issue 3, pp. 89-94, July_Sept.

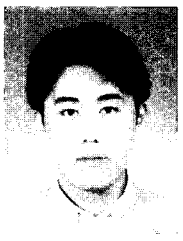
2003.
[4] P. Dobrev, D. Famolari, C. Kurzke, and B. A. Miller, "Device and Service Discovery in Home Networks with OSGi," IEEE Communications Magazine, Vol. 40, Issue 8, pp. 86-92, August 2002.
[5] K. Kang and J. Lee, "Implementation of Management Agents for an OSGi-based Residential Gateway," The 6th International Conference on Advanced Communication Technology, Vol. 2, pp. 1103-1107, 2004.
[6] F. Yang, "Design and Implement of the Home Networking Service Agent Federation Using Open Service Gateway," International Conference on Integration of Knowledge Intensive Multi-Agent Systems, pp. 628-633, Sept._Oct. 2003.
[7] H. Zhang, F. Wang, and Y. Ai, "An OSGi and Agent Based Control System Architecture for Smart Home," Proceedings of IEEE Networking, Sensing, and Control, pp. 13-18, March 2005.
[8] Knopflerfish: http://www.knopflerfish.org.
[9] K. Chen and L. Gong, *Programming Open Service Gateways with Java Embedded Server™ Technology*, Addison Wesley, 2001.
[10] L. Gong, "A Software Architecture for Open Service Gateways," IEEE Internet Computing, Vol. 5, Issue 1, pp. 64-70, Jan.-Feb. 2001.
[11] R. S. Hall and H. Cervantes, "Challenges in Building Service-Oriented Applications for OSGi," IEEE Communications Magazine, Vol. 42, Issue 5, pp. 144-149, May 2004.
[12] R. S. Hall and H. Cervantes, "An OSGi Implementation and Experience Report," First IEEE Consumer Communications and Networking Conference, pp. 394-399, Jan. 2004.

저 자 소 개

이 승 근(학생회원)
1996년 인하대학교 전자계산공학과
1998년 인하대학교 전자계산공학과 (공학석사)
2000년 인하대학교 컴퓨터정보공학과 박사수료
2000년~2005년 (주)하이켐텍 기술연구소 책임연구원
<주관심분야 : 홈네트워크, 상황인식, 웹서비스, 시멘틱웹>

김 인 태(학생회원)
1997년 인하대학교 전자계산공학과
1999년 인하대학교 전자계산공학과(공학석사)
2005년~현재 인하대학교 컴퓨터정보공학과 박사과정
<주관심분야 : 임베디드시스템, 실시간시스템>

김 태 간(학생회원)
2003년 서울산업대학교 컴퓨터공학과
2005년 인하대학교 교육대학원 정보컴퓨터교육학 석사
2005년~현재 인하대학교 컴퓨터정보공학과 박사과정
<주관심분야 : 임베디드시스템, 소프트웨어공학, 센서네트워크>



이 경 모(학생회원)
2003년 인하대학교 전기전자컴퓨터공학부
2004년~현재 인하대학교 컴퓨터정보공학과 석사과정
<주관심분야 : 정보검색, 홈네트워크, 임베디드시스템>

임 기 욱(평생회원)
1977년 인하대학교 전자공학과
1987년 한양대학교 전자계산학 석사
1994년 인하대학교 전자계산학 박사
1977년~1983년 한국전자기술연구소 선임연구원
1983년~1988년 한국전자통신연구소 시스템 소프트웨어 연구실장
1988년~1989년 미 캘리포니아 주립대학(Irvine) 방문연구원
1989년~1996년 한국전자통신연구원 시스템 연구부장, 주전산기(타이컴)Ⅲ,Ⅳ개발사업 책임자
1997년~1999년 정보통신연구진흥원 정보기술 전문위원
2001년~2003년 한국전자통신연구원 컴퓨터소프트웨어 연구소장
2000년~현재 선문대학교 컴퓨터정보학부 교수
<주관심분야: 실시간데이터베이스시스템, 운영체제, 시스템구조>

이 정 현(평생회원)
1977년 인하대학교 전자공학과
1980년 인하대학교 대학원 전자공학과(공학석사)
1988년 인하대학교 대학원 전자공학과(공학박사)
1979년~1981년 한국전자기술연구소 시스템연구원
1984년~1989년 경기대학교 교수
1989년~현재 인하대학교 컴퓨터공학부 교수
<주관심분야 : 자연어처리, HCI, 정보검색, 음성인식, 음성합성, 컴퓨터구조, 홈네트워크>