안드로이드 기반 서비스 라이브러리의 동적 재구성

이영석*

A Dynamic Service Library Reconfiguration Method of Android Platform

Young-seok Lee*

요 약

Android 모바일 플랫폼은 플랫폼에 설치된 모든 어플리케이션들의 클래스(class) 파일들을 복사하여 하나의 디렉터리에 보관하고, 해당 디렉터리에서 메모리 맵 파일(memory-mapped file) 방식으로 참조함으로써 서비스 라이브러리를 플랫폼에 추가하기 위해서는 플랫폼의 바이너리(binary) 이미지를 반드시 새롭게 생성하여 플랫폼을 교체해야만 한다. 또한, 이러한 메모리 맵 파일 방식의 참조는 어플리케이션이 참조하는 라이브러리가 중복될 경우 해당 디렉터리에서도 중복되어 포함되므로 디바이스의 저장 공간 낭비를 불러오게 된다. 따라서, Android 모바일 플랫폼상에서 동작하는 각 어플리케이션들의 서비스 라이브러리를 공유 라이브러리 형태로 관리하며, 플랫폼의 버전 관리 및 기능 추가 및 변경을 실시간 및 동적으로 지원할 수 있는 Android 플랫폼용 동적 재구성 기법이 필요하다. 본 논문에서는 Android 모바일 플랫폼을 대상으로 하여, 동적으로 서비스 라이브러리를 추가할 수 있는 플랫폼 동적 재구성 기법을 제안한다.

ABSTRACT

Android mobile platform copy the class files of all applications installed in android platform and keep them into one directory. But to add the service library into a platform, Android must generate a new binary image of platform and replace the platform because android refer the libraries using the memory-mapped file method in a directory. And this memory-mapped file reference method waste the storage space of mobile device because that directory include the duplicated libraries if the libraries referred by application are duplicated, Therefore a dynamic reconfiguration method on android mobile platform is needed to manage the service libraries of application executed on android platform and support the version management and function addition/modification. In this paper, we propose the platform dynamic reconfiguration method to add dynmically the service libraries on android platform.

키워드

안드로이드, 모바일 플랫폼, 서비스 라이브러리

Key word

Android, Mobile Platform, Service Library

* 종신회원 : 군산대학교 정보통신공학과(leeys@kunsan.ac.kr)

접수일자 : 2012. 12. 03

심사완료일자 : 2013. 01. 03

I. 서 론

일반적으로 모바일 어플리케이션은 모바일 플랫폼에서 제공하는 API를 이용하여 개발된다[1]. 그러나 홈네트워킹 서비스나, 증강 현실 서비스를 제공하는 라이브러리와 같이 기본 모바일 플랫폼에 포함되지 않은 라이브러리들도 어플리케이션 개발에 많이 사용된다[2,3]. Android 모바일 플랫폼에서도 이와 같은 외부 라이브러리들은 어플리케이션 코드의 일부가 되어 어플리케이션 내에 포함된다.

Android 모바일 플랫폼은 플랫폼에 설치되는 모든 어플리케이션들의 라이브러리를 포함하는 클래스(class) 파일들을 복사하여 하나의 디렉터리에 보관하고, 각 어플리케이션은 필요한 클래스 파일을 해당 디렉터리에서 메모리 맵 파일(memory-mapped file) 방식으로 참조하게 된다. 따라서, 각 어플리케이션에서 동일한 라이브러리를 중복하여 참조하게 될 경우 해당 디렉터리에도 중복해 포함하게 되므로 디바이스의 저장 공간 낭비를 불러오게 된다. 그러므로 추가하고자 하는 라이브러리를 모바일 플랫폼에 공유 라이브러리 형태로 추가하여 여러 어플리케이션이 이를 공유해서 사용할 수 있는 방법이 필요하다. 그러나 현재 Android 모바일 플랫폼에서 새로운 라이브러리를 공유 라이브러리 형태로 추가하기 위해서는 기존의 플랫폼에 새롭게 추가할 라이브러리를 포함하는 바이너리(binary) 이미지를 새로 생성함으로써 모바일 단말기의 플랫폼을 교체해야만 한다. 그러나 이와 같은 오프라인 형태의 서비스 방식은 불편하고 매우 비효율적이므로 모바일 단말기 상에 동작 중인 플랫폼에 동적으로 라이브러리를 추가할 수 있는 플랫폼 동적 재구성 기법 즉 동적 기능 확장, 축소 기법이 필요하다. 그러나 지금까지 플랫폼 동적 재구성 기법에 대한 연구들이 활발히 진행되어 왔지만[4], 모바일 플랫폼을 대상으로 한 동적 재구성 기법에 대한 연구는 많지 않다[5,6]. 특히 Android 모바일 플랫폼을 대상으로 한 연구는 본 연구가 처음이다.

본 논문에서는 Android 모바일 플랫폼을 대상으로 하여, 플랫폼에 동적으로 라이브러리를 추가할 수 있는 플랫폼 동적 재구성 기법을 제안한다. 또한, 제안하는 플랫폼 동적 재구성 기법을 상용 Android 모바일 단말기에 적용하고자 할 때 Android 보안 아키텍처와 단말기 제조사의 보안 정책으로 인해 발생할 수 있는 문

제점을 분석하여 이를 해결하기 위한 새로운 방안을 제안한다.

II. 관련 연구

2.1. 동적 재구성 기법

어플리케이션이나 시스템은 여러 개의 개체(entity)들로 구성되는데 이때, 개체는 오브젝트(object), 클래스(class), 컴포넌트(component) 등이 될 수 있다. 재구성이란 이러한 개체들을 추가, 삭제 혹은 갱신하는 것을 말한다. 재구성 기법은 크게 정적인 재구성(static reconfiguration) 기법과 동적인 재구성(dynamic reconfiguration) 기법으로 나뉘어 진다.

정적 재구성은 시스템의 실행을 중단한 후 오프라인으로 시스템 재구성을 수행하는 반면, 동적 재구성은 시스템의 중단 없이 시스템의 재구성이 가능하여 변화된 환경이나 사용자의 요구에 따라 대응할 수 있다는 장점이 있다[7]. 이러한 동적 재구성 기법에 대한 연구로는 MMLite 운영체제의 Mutation 기법[2], K42 운영체제의 온라인 재구성 기법[3], Kea 운영체제의 동적 재구성 기법[6], SDR(Software Defined Radio) 시스템[8]의 통신 프로토콜 스위칭(switching)을 위한 동적 재구성 기법[9] 등이 있다. 그러나 [2,3]은 모바일 플랫폼을 대상으로 하는 연구가 아니며, [10]은 모바일 플랫폼을 대상으로 하지만 무선 통신 기술인 SDR을 지원하기 위한 동적 재구성 기법이다. 그러므로 모바일 플랫폼의 기능을 확장할 수 있는 동적 재구성 기법을 제안하는 본 논문과는 차이가 있다.

2.2. 안드로이드 플랫폼

Android 플랫폼은 운영체제와 미들웨어 그리고 기본 어플리케이션을 포함하고 있는, 모바일 디바이스를 위한 플랫폼이며, 그 구조는 그림 1과 같다[11].

그림 1에서 Linux Kernel 계층은 보안, 메모리 관리, 프로세스 관리, 네트워크 프로토콜 스택 및 드라이버 모델과 같은 핵심 시스템 서비스들로 구성되어 있다. Libraries 계층은 Linux Kernel이 제공하는 시스템 콜(system call)을 이용하여 개발된 라이브러리들을 포함하고 있으며, 이 계층의 라이브러리들은 Application Framework 계층을 통해 어플리케이션에 제공된다.

그림 1. Android 플랫폼 구조
Fig. 1 Architecture of Android Platform

Application Framework 계층은 어플리케이션이 동작하기 위해 필요한 요소 서비스들과 하드웨어와의 인터페이스를 담당하는 하드웨어 서비스들을 어플리케이션에 제공한다. Application Framework 계층의 라이브러리들은 Java 언어로 개발되어 있다. 이에 비하여 Libraries 계층의 라이브러리들은 C/C++ 언어로 개발되어 있으며 JNI(Java Native Interface)를 통해 Application Framework 계층과 연결되어 어플리케이션에 제공된다.

III. 안드로이드 기반 동적 재구성 기법

그림 2에 표시된 것처럼 Android 플랫폼에 포함되지 않은 라이브러리(회색 표시)를 Libraries 계층과 Framework 계층에 추가할 수 있다면 어플리케이션에 더 많은 기능을 제공할 수 있을 것이다.

그림 2. 외부 라이브러리가 추가된 안드로이드 플랫폼
Fig. 2 External Library Addition in Android Platform

또한, 라이브러리를 추가하기 위해 Android 플랫폼의 소스를 수정하고 플랫폼 이미지를 새로 생성하는 방법을 사용하지 않고 모바일 단말기에 탑재된 Android 플랫폼에 동적으로 라이브러리를 추가할 수 있다면 더 효율적일 것이다. 본 장에서는 Application Framework 계층과 Libraries 계층에 새로운 라이브러리를 동적으로 추가하여 플랫폼의 기능을 동적으로 확장할 수 있는 방법을 제안한다.

3.1. 개발 환경 구성

동적으로 추가될 라이브러리가 제공하는 기능을 어플리케이션이 이용하기 위해서는 어플리케이션 개발 단계에서 해당 라이브러리가 제공하는 API를 이용하여 어플리케이션이 개발되어야 한다. 그렇게 하기 위해서는 Android 플랫폼에 새롭게 추가할 라이브러리를 이용하여 어플리케이션을 개발할 수 있는 개발 환경을 제공하는 방법에 대한 연구가 선행되어야 한다.

Android 개발 환경은 Google service와 data를 이용할 수 있게 해주는 external 라이브러리를 Android SDK (Standard Development Kit)에 추가하여 사용할 수 있게 해주는 Google APIs Add-On 기능을 제공한다. 본 절에서는 Google APIs Add-On 기능을 이용하여 Android SDK에 새로운 라이브러리를 추가함으로써, 추가된 라이브러리를 이용하는 어플리케이션을 개발할 수 있는 개발 환경을 구성하는 방법에 대해 기술한다.

표 1. add-on 라이브러리 구성 항목
Table. 1 Add-on Library Composition

항목	설명
images	기본 Android 플랫폼에 external 라이브러리가 추가된 image 파일. 에뮬레이터에서 사용
libs	external 라이브러리 파일
manifest.ini	라이브러리의 name, vendor, version 명세. package명과 연결된 file명 명세

표 1은 Google APIs Add-on 기능을 이용하여 Android SDK에 추가되는 add-on 라이브러리의 주요 구성 항목을 나타낸다. 표 1의 세 항목을 생성하여 Android SDK에 새로운 라이브러리를 추가하고, 이를 이용하는 어플리케이션을 개발할 수 있는 개발 환경을 구성한다.

3.2. 새로운 라이브러리 추가

라이브러리가 추가될 때마다 Android 소스를 수정하여 플랫폼 image를 재생성하여야 한다면 매우 비효율적이며, 기존에 사용되고 있는 스마트폰에 새로운 플랫폼 image를 배포하는 작업 또한 사용자에게 큰 불편함을 주게 될 것이다. 본 절에서는 모바일 단말기에 설치된 Android 플랫폼 상의 Application Framework 계층과 Libraries 계층에 동적으로 새로운 라이브러리를 추가하는 방법에 대하여 기술한다.

3.2.1. Application Framework 계층 라이브러리

Android 어플리케이션과 Application Framework 계층의 라이브러리는 JAVA 언어로 작성되지만 일반 JAVA virtual machine이 아닌 모바일 단말기에 적합한 Dalvik virtual machine 위에서 동작한다.

그러므로 JAVA byte code로 컴파일된 어플리케이션과 Application Framework 계층의 라이브러리들은 dalvik byte code인 dex(dalvik executable) 파일 형태로 변환해야 한다. 또한, Android 플랫폼이 제공하는 표준 API가 구현된 파일은 dex 파일을 포함하는 jar 파일 형태로 모바일 단말기의 파일 시스템에 적재되므로 dex 파일 형태로 생성한 라이브러리를 최종적으로 jar 파일로 변환한 후 모바일 단말기에 적재해야 한다. 이러한 변환 작업을 위해 Android SDK에서 dx 툴을 제공한다. dx 툴을 이용하여 생성한 Android 플랫폼용 라이브러리를 모바일 단말기의 파일 시스템에 추가한 후 3.3절에서 설명할 라이브러리 등록 과정을 마치면 Application Framework 계층에 라이브러리를 동적으로 추가할 수 있다.

3.2.2. Libraries 계층 라이브러리

Android가 제공하는 Libraries 계층의 라이브러리들은 C/C++ 언어로 구현되어 있으며 JNI(Java Native Interface)를 통해 Application Framework 계층과 연결되어 어플리케이션에 제공된다. 이와 같은 라이브러리들은 모바일 단말기 파일 시스템의 /system/libs/ 디렉터리에 .so 파일 형태로 존재한다. Android 개발 환경은 JNI를 이용하여 C/C++ 기반의 코드를 사용하는 어플리케이션을 개발할 수 있도록 NDK(Native Development Kit)를 제공한다. 이를 이용하여 C/C++ 기반의 라이브러리를 생성한 후 /system/libs/ 디렉터리에 추가하고 이와 JNI

로 연결하는 Java class를 3.2.1절의 방법을 이용하여 Application Framework 계층에 추가하면 기존의 라이브러리들과 동일한 메커니즘으로 어플리케이션에 기능을 제공할 수 있다.

3.3. 라이브러리 등록

어플리케이션이 Android 플랫폼에서 제공하는 표준 API 외의 add-on 라이브러리를 사용할 경우 어플리케이션의 manifest 파일에 해당 라이브러리에 대한 정보를 아래와 같이 명시해 주어야 한다.

```
<uses-library android:name="라이브러리 이름" />
```

Android 플랫폼의 'Package Installer'는 어플리케이션 설치 시, 어플리케이션의 manifest 파일을 분석하여 add-on 라이브러리를 사용하는지 검사하고, 사용할 때에는 해당 라이브러리가 플랫폼에 존재하는지 검사한다. 이를 위해 Android 플랫폼은 플랫폼에 설치된 라이브러리의 정보를 /system/etc/permissions/ 디렉터리에 xml 파일 형태로 관리하고 시스템 부팅 시, xml 파일로부터 정보를 추출하여 메모리에 유지한다. /system/etc/permissions/ 디렉터리에서 라이브러리 정보를 관리하는 xml 파일의 내용은 다음과 같다.

```
<library name="라이브러리 이름(package 이름)"
file="파일 경로 및 이름" />
```

위와 같이 라이브러리의 이름과 해당 라이브러리가 구현된 파일의 위치 및 이름을 명시한 xml 파일을 /system/etc/permissions/ 디렉터리에 추가한 후 시스템을 재부팅하면 해당 라이브러리에 대한 정보를 메모리에 유지하게 되고 어플리케이션이 해당 라이브러리를 이용할 수 있다.

앞서 기술한 Android 플랫폼 동적 재구성 기법을 이용하여 모바일 단말기에 탑재된 Android 플랫폼에 동적으로 라이브러리를 추가하는 어플리케이션을 생성하여 배포한다면 단말기 사용자는 쉽게 라이브러리를 추가할 수 있을 것이다.

IV. 문제점 분석 및 Library Installer를 통한 해결 방안

3장에서 기술한 Android 플랫폼 동적 재구성 기법은 기술적 측면에서 실현 가능하지만 이것을 상용 Android 모바일 단말기 상에 적용할 때에는 중대한 문제가 발생한다. 본 절에서는 이러한 문제점을 기술하고, 이에 대한 해결 방안으로 Library Installer를 통한 해결 방안을 추가적으로 제안한다.

4.1. 문제점

대부분의 상용 Android 모바일 단말기기는 파일 시스템의 중요 디렉터리 및 파일에 대한 user id와 group id를 root로 설정하고 있으며, root 이외의 user id에는 read 권한만 부여하도록 접근 권한이 설정되어 있다. 모든 Android 어플리케이션은 설치될 때 시스템으로부터 자신의 고유한 user id를 할당받는데 어플리케이션이 할당받는 user id는 root가 될 수 없기 때문에 어플리케이션은 user id가 root로 설정된 디렉터리 및 파일에 대해 read 권한만 가질 수 있게 된다.

이는 사용자 혹은 어플리케이션이 고의 혹은 실수로 파일 시스템의 중요 부분을 변경하여 단말기가 오작동하는 것을 방지하기 위한 단말기 제조사의 정책이다. 따라서, 3장에서 제안한 Android 플랫폼 동적 재구성 기법을 이용하기 위해서는 user id가 root로 설정된 디렉터리에 파일을 추가해야 한다. 그러므로 3장에서 제안한 동적 재구성 기법은 상용 모바일 단말기에서는 정상적으로 동작하지 못하며 이를 해결할 수 있는 방안이 필요하다.

4.2. 해결 방안

모든 Android 어플리케이션은 디지털 키 인증서로 서명되어야 하며, 일반적으로 어플리케이션을 배포하기 전 어플리케이션 개발자가 자신이 생성한 개인 키로 서명한다. 이러한 서명 작업은 어플리케이션 간의 신뢰 관계를 확립하기 위해 사용되며, 어플리케이션의 권한 요청과 user id 공유 요청에 대한 허가 여부를 결정하는데 사용한다. Android 어플리케이션은 자신의 manifest 파일에 'sharedUserId' 항목을 명시하여 다른 어플리케이션 혹은 시스템 자원과 동일한 user id를 공유할 수 있는데, 이 때 플랫폼에 포함된 키 중 'platform' 키로 서명한다.

어플리케이션은 'system' user id를 공유할 수 있다. Android 플랫폼은 서로 다른 용도의 네 가지의 키를 포함한다(표 2).

표 2. Android 플랫폼에 포함되는 키
Table. 2 Key included in Android Platform

이름	설명
platform	핵심 패키지를 위한 키
shared	Home/contacts 관련 패키지를 위한 키
media	Media/download 관련 패키지를 위한 키
releasekey	기본 키

단말기 제조사는 제조사 고유한 표 2에서와 같은 네 가지의 키를 생성하여 플랫폼에 추가한 후, 플랫폼 이미지를 생성하며, 플랫폼에 포함되어 있는 built-in 어플리케이션들은 앞서 기술한 네 가지의 키 중 하나의 키로 서명되어 플랫폼 이미지에 포함되게 된다. 플랫폼에 포함되어 플랫폼이 설치될 때 함께 설치되는 built-in 어플리케이션의 경우 단말기 제조사가 추가하는 것이므로 제조사가 생성한 'platform' 키를 이용해 서명할 수 있으므로 'system' user id를 공유할 수 있다. 그러나 일반적인 어플리케이션의 경우 서명에 사용된 키가 모바일 단말기 제조사가 생성한 'platform' 키와 다르기 때문에 'system' user id를 공유할 수 없다. 'system' user id를 공유하는 어플리케이션은 일반 어플리케이션보다 더 많은 권한을 부여 받을 수 있다.

본 절에서는 위와 같은 Android 플랫폼의 특성을 이용하여 상용 모바일 단말기 상에서 플랫폼을 동적으로 재구성할 수 있는 방안을 제안한다. 제안하는 Android 플랫폼 동적 재구성 기법은 다음과 같다.

즉, 단말기 제조사가 'system' user id를 사용하는 Library Installer를 built-in 어플리케이션으로 제공하고 외부 라이브러리가 추가될 디렉터리와 /system/etc/permissions/ 디렉터리의 user id를 'system'으로 설정하여 플랫폼 이미지를 제공한다면 본 논문에서 추가적으로 제안하는 "Library Installer"를 이용하여 상용 모바일 단말기 상에 동적으로 라이브러리를 추가할 수 있다.

1. 추가 라이브러리가 저장될 디렉토리를 생성 후 user id를 'system'으로 설정
2. /system/etc/permissions/ 디렉터리의 user id를 'root'에서 'system'으로 변경
3. 위의 두 디렉터리에 대해 'system' user 이외의 user에 대한 접근 권한을 'read' 권한으로 설정
4. 'system' user id를 사용하는 라이브러리 설치 어플리케이션(Library Installer)을 모바일 단말기 제조사가 built-in으로 제공

“Library Installer”를 이용함으로써, 플랫폼에 새롭게 추가되는 라이브러리 파일을 관리하기 위해 Android 플랫폼의 기본 라이브러리를 관리하는 디렉터리(/system/framework/, /system/libs/)를 사용하지 않고 별도의 디렉토리를 추가하여 사용함으로써 기본 라이브러리에 대한 보안을 유지할 수 있다. 또한, 추가되는 라이브러리 파일을 관리하기 위한 디렉터리가 'system' user id를 사용하고 'system' user 이외의 user에게는 'read' 권한만 부여하므로 추가되는 라이브러리들을 일반 어플리케이션으로부터 보호할 수 있다. user id를 'root'에서 'system'으로 변경한 /system/etc/permissions/ 디렉터리도 신뢰할 수 있는 “Library Installer”에게만 추가적으로 접근 권한을 허용해주는 것이므로 'root' user id를 사용할 때와 마찬가지로 보안을 유지할 수 있다. 그러므로 본 제안하는 동적 재구성 기법을 상용 모바일 단말기에 적용하더라도 단말기 제조사 입장에서 시스템 보안에 관한 문제가 발생하지 않는다. 본 논문에서 제안하는 “Library Installer”를 포함하는 동적 재구성 기법을 위한 Android 플랫폼의 구조는 그림3과 같다.

그림 3. 동적 재구성이 가능한 Android 플랫폼 구조
Fig. 3 Dynamic Reconfiguration Android Architecture

“Library Installer”는 Applications 계층에서 동작하는 어플리케이션으로서, 좀 더 효율적인 동적 재구성을 위해 다음과 같은 내부 기능 모듈들을 포함하도록 설계 및 구현하였다(그림 4).

그림 4. Library Installer 구조
Fig. 4 Library Installer Architecture

- 검사기(Checker): 설치 요청을 받은 라이브러리가 이미 설치되어 있는지 여부를 검사한다.
- 분석기(Analyzer): 라이브러리 파일을 분석하여 /system/etc/permissions/ 디렉터리에 추가할 라이브러리 정보 파일을 생성한다. 3.3절에서 설명한 것과 같이 라이브러리 정보 파일에는 라이브러리의 이름과 라이브러리 파일의 위치 및 이름에 대한 정보가 명시되어야 하는데 파일의 위치는 라이브러리가 설치될 모바일 단말기에서 결정되므로 모바일 단말기에서 라이브러리 정보 파일을 생성해야 한다.
- 다운로더(Downloader): 라이브러리 서버(Library Server)로부터 라이브러리의 다운로드를 수행하는 모듈이다. 라이브러리 서버는 일반적인 파일 서버로서 Android Market이 될 수도 있고 포탈 서버일 수도 있다.
- 저장 관리자(Storage Manager): 라이브러리 파일들을 백업(backup)하고 문제가 발생했을 때 복구(restore)하는 모듈이다.

V. 구현 및 시험

제안하는 Android 플랫폼 동적 재구성 기법을 검증하기 위해 외부 라이브러리를 이용하는 어플리케이션을 개발할 수 있는 개발 환경과, 라이브러리를 제공하는 외

부 서버로부터 라이브러리를 다운로드 받아 Android 플랫폼에 설치하는 'Library Installer' built-in 어플리케이션을 구현하였다. 또한, 사칙연산 함수를 제공하는 'calc' 라이브러리와 이 라이브러리를 이용하는 계산기 어플리케이션을 구현하였다.

5.1. 구현 및 시험 환경

라이브러리 설치 프로그램의 구현 및 시험 환경은 다음과 같다. 개발 툴로는 eclipse 3.5.1과 ADT(Android Development Tools) 0.9.7 플러그인을 사용하였으며, Android SDK 2.1 버전을 사용하였다. 라이브러리를 제공하는 서버는 Apache Tomcat 6.0을 이용하여 구축하였다. 실험은 Android 플랫폼 2.1 버전이 탑재된 Odroid-T 개발 보드 상에서 수행하였다.

5.2. 개발 환경 구현

add-on 라이브러리의 구성 항목인 libs 디렉터리에 'calc' 라이브러리 구현 파일을 추가하고, 'calc' 라이브러리의 manifest 파일은 그림 5와 같이 작성한 후 추가하였다.

```
# SDK Add-on Manifest

name=Distsys APIs
vendor=Distsys lab.
description=Android + Distsys APIs

# version of the Android platform on which this add-on is built.
api=7

# revision of the add-on
revision=1

# list of libraries, separated by a semi-colon.
libraries=android.example

# details for each library
android.example=calc.jar;
```

그림 5. 'calc' 라이브러리의 manifest.ini 파일
Fig. 5 'calc' 라이브러리의 manifest.ini 파일

Android 프로젝트 생성 시, 플랫폼 선택 메뉴에 'calc' 라이브러리를 이용할 수 있는 플랫폼이 추가되고, 'calc' 라이브러리의 manifest 파일에 명시한 정보가 표시되는 것을 확인할 수 있다(그림 6).

그림 6. Android project 선택 화면
Fig. 6 Android project Window

'calc' 라이브러리를 이용할 수 있는 플랫폼을 선택하여 Android 프로젝트를 생성하면, Android SDK에서 제공하는 표준 API 외에 'calc' 라이브러리가 어플리케이션이 이용할 수 있는 API 목록에 추가된 것을 확인할 수 있다(그림 7).

그림 7. 추가된 라이브러리
Fig. 7 Library Adding

추가된 라이브러리를 이용하여 어플리케이션을 개발할 수 있는지 검증하기 위해 'calc' 라이브러리를 이용하는 계산기 어플리케이션을 개발하였다.

5.3. Library Installer 어플리케이션

Odroid-T 개발보드에 탑재할 Android 플랫폼 이미지를 다음과 같이 생성하였다. 먼저 구현한 "Library Installer" 어플리케이션 소스와 임의로 생성한 플랫폼 키를 Android 플랫폼 소스에 추가한 후 빌드하였다. 그 다음 빌드 후 생성된 어플리케이션 설치 파일을 플랫폼 소스에 추가하였던 플랫폼 키로 서명하였다. 마지막으로, 추가 라이브러리를 관리하기 위한 '/system/extendedLib/' 디렉터리를 생성하고 이 디렉터리와 '/system/etc/per

issions/' 디렉터리의 user id를 system으로 설정한 후 플랫폼 이미지를 생성하였다. 이렇게 생성한 Android 플랫폼 이미지를 Odroid-T 개발보드에 탑재한 후 "Library Installer" 어플리케이션을 실행시키면 다른 어플리케이션과 달리 system user id로 동작하는 것을 확인할 수 있다(그림 8).

그림 8. Library Installer의 user id
Fig. 8 Library Installer user id

"Library Installer" 어플리케이션이 정상적으로 동작하는지 확인하기 위해 "Library Installer" 어플리케이션을 통해 'calc' 라이브러리를 설치하기 전과 후의 계산기 어플리케이션의 동작을 비교하였다.

계산기 어플리케이션의 manifest 파일에 'uses-library' 항목을 명시하였을 경우에는 'calc' 라이브러리가 설치되기 전에 계산기 어플리케이션이 설치되지 않는다. 'uses-library' 항목을 명시하지 않았을 경우에는 계산기 어플리케이션이 설치되는 하지만 계산기 어플리케이션을 실행하면 정상적으로 동작하지 않는다(그림 9).

그림 9. 라이브러리 설치 전 어플리케이션 설치 및 실행
Fig. 9 Application Install and Execution before installing Library

"Library Installer" 어플리케이션을 통해 'calc' 라이브러리를 설치하면 '/system/extendedLib/' 디렉터리에 'calc' 라이브러리 구현 파일이 추가되며 '/system/etc/permissions/' 디렉터리에 라이브러리 설치 위치 정보가 담긴 xml 파일이 추가된다. Library Installer 어플리케이션에 의해 'calc' 라이브러리가 설치된 후 계산기 어플리케이션이 정상적으로 동작한다(그림 10).

그림 10. 라이브러리 설치 후 어플리케이션 설치 및 실행
Fig. 10 Application Install and Execution After installing Library

VI. 결 론

본 논문에서는 Android 모바일 플랫폼에 동적으로 라이브러리를 추가하고, 이를 어플리케이션이 이용할 수 있게 해주는 플랫폼 동적 재구성 기법을 기술하였다. 또한, 상용 Android 모바일 단말기에 적용할 때 발생할 수 있는 문제점과 이에 대한 해결 방안 및 구현 내용을 기술하였다.

본 논문에서 제시한 플랫폼 동적 재구성 기법은 플랫폼의 기능을 확장하기 위해 플랫폼 이미지를 재생성하는 작업이 필요하지 않으며 모바일 단말기에 탑재된 플랫폼 상에 라이브러리를 추가하여 동적으로 플랫폼의 기능을 확장할 수 있다. 또한, Android 플랫폼의 보안 정책과 모바일 단말기 제조사의 보안 정책에도 지장이 없는 효과적인 방법이다. 향후 연구로는, 본 논문에서 제안한 모바일 플랫폼 동적 재구성 기법의 LiMo나 bada와 같은 다양한 개방형 플랫폼에 대한 적용성을 검증할 예정이다.

참고문헌

- [1] Google Android, <http://www.android.com/>, Accessed on Sep. 1, 2010.
- [2] LiMo Foundation, <http://www.limofoundation.org/>, Accessed on Sep. 1, 2010.
- [3] Samsung bada, <http://www.bada.com/>, Accessed on Sep. 1, 2010.
- [4] Johannes Helander, Alessandro Forin, "MMLite: A Highly Componentized System Architecture," Eighth ACM SIGOPS European Workshop, Sep. 1998.
- [5] Craig A. N. Soules, Jonathan Appavoo, Kevin Hui, Robert W. Wisniewski, Dilma DaSilva, Gregory R. Ganger, Orran Krieger, Michael Stumm, Marc Auslander, Michal Ostrowski, Bryan Rosenburg, Jimi Xenidis, "System Support for Online Reconfiguration," Proceedings of the 2003 USENIX Technical Conference, pp.141-154, 2003.
- [6] 송인준, 김영필, 유혁, "휴대 기기에서의 SDR (Software Defined Radio)을 위한 운영체제 구조와 재구성 기법", 한국정보과학회 가을 학술발표논문집, 제30권 제2호, pp.148-150, 2004. 4.
- [7] 박충범, 이수원, 최훈, "웨어러블 컴퓨터를 위한 Self-Configuration Manager", 한국정보과학회 KCC2006 추계학술발표회 논문집, 제33권 제2호, pp.233-237, 2006.10.
- [8] 이수원, 유인선, 유용덕, 최훈, "컴포넌트 기반 소프트웨어 플랫폼을 위한 컴포넌트 관리자 설계 및 구현", 한국정보과학회 KCC2006 춘계학술발표회 논문집, 제33권 제1호, pp.199-201, 2006.06.
- [9] 권정호, 김재훈, "컴포넌트기반 미들웨어에서 효율적인 컴포넌트 재구성", 한국컴퓨터종합학술대회 2005 논문집 Vol.32, No.1(A), pp.751-753, 2005.
- [10] Xuejun Chen, "Dependence management for dynamic reconfiguration of component-based distributed systems," Proceedings of ASE 2002, pp.279-284, Sep. 2002.
- [11] SDR Forum, <http://www.sdrforum.org/>, Accessed on Sep. 1, 2010.

저자소개

이영석(Young-seok Lee)

1992년 충남대학교
컴퓨터공학과 공학사
1994년 충남대학교
컴퓨터공학과 공학석사

2002년 충남대학교 컴퓨터공학과 공학박사
1994년~1997년 LG전자정보통신연구소 연구원
2002년~2004년 한국전자통신연구원 선임연구원
2011년~2012년 사우스플로리다대학교 방문교수
2004년~현재 군산대학교 정보통신공학과 부교수
※ 관심분야: 정보보호, 이동컴퓨팅, 컴퓨터네트워크