JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 18, No. 1 : 50~56 Jan. 2014

안드로이드 플랫폼의 GPS 위치 제공자에 대한 동작 분석

이계상*

An Analysis of the Operation of the GPS Location Provider in the Android Platform

Kyesang Lee*

Department of Information and Communications Engineering, Dong-eui University, Busan 614-714, Korea

요 약

최근 위치 기반 서비스를 사용하는 안드로이드 앱이 널리 사용되고 있다. 안드로이드 플랫폼에서 지원되는 여러 위치 제공자 중 GPS (Global Positioning System) 위치 제공자의 동작에 대한 이해는 관련 안드로이드 개발자에게 중요한 과제이다. 본 고는 안드로이드 플랫폼의 GPS 서브시스템 소스를 분석하여, GPS 위치 제공자의 동작을, 주요 쓰레드를 중심으로 그리고 초기화 단계부터 최종 위치 보고 단계까지 단계별로 일목요연하게 제시하고자 한다.

ABSTRACT

Recently Android apps utilizing location based services are widely used. Understanding the operation of the GPS location provider, among various location providers supported in the Android platform, is an essential task for developers using the Android. This paper, based on the analysis of source codes of the GPS subsystem in the Android platform, shows clearly and orderly the operation of the GPS location provider, in key threads' perspectives as well as in a stepwise fashion from the initialization up to the final location report steps.

키워드 : 안드로이드 플랫폼, GPS, 위치 제공자, 소스 분석, 위치 기반 서비스

Key word : Android platform, GPS, location provider, source analysis, location based service

접수일자 : 2013. 11. 05 심사완료일자 : 2013. 11. 28 게재확정일자 : 2013. 12. 13

* **Corresponding Author** Kyesang Lee(E-mail:ksl789@gmail.com, Tel:+82-51-890-1691)
Department of Information and Communications Engineering, Dong-eui University, Busan 614-714, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2014.18.1.50 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

최근 안드로이드 앱의 대부분은 위치 기반 서비스를 포함한다. 안드로이드 시스템은 여러 위치 제공자를 지원한다. GPS 기반, 와이파이 기반, 그리고 기지국 기반의 위치 제공자가 대표적이다. GPS 위치 제공자는 실외에서 가장 보편적으로 사용되며 상당히 정확한 위치를 알려주기 때문에 널리 사용되고 있다.

안드로이드 플랫폼에서 GPS 위치 제공자가 어떻게 동작되는지에 대한 이해가 위치 기반 서비스 앱을 개발하는 개발자와 안드로이드 플랫폼을 맞춤화하는 개발자에게 매우 중요한데 비해서, 방대한 소스를 읽고 이해하기는 어렵고, 관련 분석 연구 결과는 거의 없는 편이다. 참고 문헌[1-2]은 안드로이드의 GPS 위치 제공자를 소스 코드 레벨에서 함수 호출 흐름을 따라 분석하였다. 하지만, 함수 호출을 그대로 따라가며 세부 코드를 제시함으로써, 주요 쓰레드의 기능 등 GPS 서브시스템의 체계적인 구조를 쉽게 파악하기가 힘들다.

본 고는 소스 코드[3] 분석을 토대(표 1 참조)로 하지만, 함수 호출 코드를 보여 주는 대신, GPS 위치 제공자를 구성하는 주요 쓰레드를 중심으로 하여, 초기화부터 최종 위치 보고까지 단계별로, 각 쓰레드의 위치 정보 제공 과정에서의 역할을 다이어그램과 함께 제시하였다. 본 고를 통해, 안드로이드에서 GPS 위치 제공자의 동작을 쉽고 빠르게 이해할 수 있으리라 기대한다.

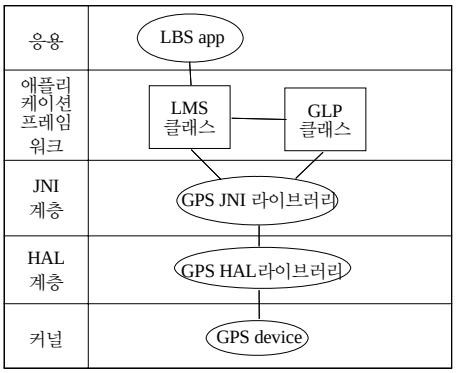

그림 1. GPS 서브시스템의 구성
Fig. 1 Components of GPS subsystem

본론에 앞서, 그림 1은 GPS 서브시스템의 구성을 안드로이드 계층구조에 맞춰 보인다. 애플리케이션 프레임워크 계층에 Location Manager Service (LMS) 클래스, GPS Location Provider (GLP) 클래스가 구현되며, 라이브러리 계층은 JNI (Java Native Interface)와 HAL (Hardware Abstraction Layer) 라이브러리로 나뉜다. GPS 장치는 커널에 하나의 장치 파일로 적재된다.

Ⅱ. GPS 위치 제공자의 로딩 및 초기화

이 장은 System Server (SS) 프로세스의 시작으로부터 GPS 위치 제공자의 로딩 및 초기화까지의 과정을 기술한다. SS 프로세스는 안드로이드 시스템이 부팅 후 최초로 실행시키는 자바 응용 프로그램이다. 이 프로세스는 네이티브 시스템 서비스와 자바 시스템 서비스를 기동시킨다.

그림 2와 같이, SS 프로세스는 먼저, 시스템 서비스 실행에 필요한 각종 JNI 네이티브 함수를 포함하는 '서버 라이브러리'를 메모리에 로드한다. 이 중, GPS 서비스를 위한 네이티브 함수들(GPS JNI 라이브러리)도 함께 로드된다. 라이브러리를 로드한 후, SS 프로세스는 네이티브 시스템 서비스를 먼저 기동시킨다. 다음, 자바 시스템 서비스를 기동시키기 위해 Server 쓰레드를 생성하여 실행시킨다.

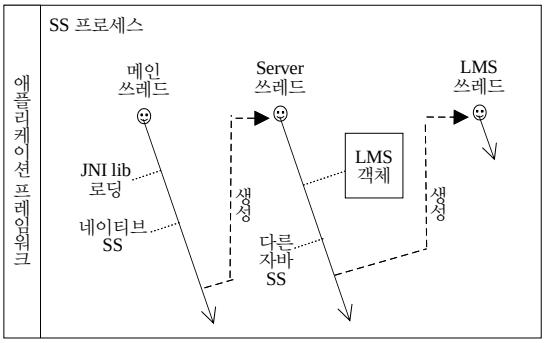

그림 2. LMS 서비스의 시작
Fig. 2 Start of LMS service

2.1. Server 쓰레드와 LMS 서비스 시작

Server 쓰레드는 다양한 자바 시스템 서비스를 기동시킨다. 그 중 하나로 Location Manager Service (LMS) 서비스가 그림 2와 같이 시작된다. 즉, Server 쓰레드는 LMS 클래스 객체를 생성하고, 이를 Service Manager

를 통해 Context Server에 등록한다. 그리고, LMS 쓰레드를 생성하여 실행시킨다.

2.2. LMS 쓰레드

LMS 쓰레드는 여러 위치 제공자를 로딩하고, 위치 제공자들이 제공하는 위치 정보를 수신하여 이를 요청한 클라이언트 응용 프로그램들에 제공한다. 본 고는, LMS 쓰레드와 GPS 위치 제공자 간의 상호 작용 분석에 집중한다.

LMS 쓰레드는 Server 쓰레드에 의해 실행될 때, 그림 3과 같이, 메시지 핸들러를 준비하고, 초기화 과정으로 GPS 위치 제공자의 로딩을 수행한다. 메시지 핸들러는 GPS 위치 제공자로부터 위치 정보를 수신할 때 이용된다.

2.3. LMS 쓰레드의 GPS 위치 제공자 로딩

LMS 쓰레드의 GPS 위치 제공자 로딩은 그림 3과 같이, 먼저, GPS 인터페이스가 지원되는지를 확인하는 과정으로 시작된다. 이 확인은 GPS Location Provider (GLP) 클래스의 JNI 네이티브 함수로 구현되며 그 내용은 다음과 같다. 이 함수를 포함하는 'GPS JNI 라이브러리'는 앞에서 언급한 바와 같이 '서버 라이브러리' 로드 때 이미 SS 프로세스에 의해 로드된 상태이다.

그림 3과 같이, 우선, HAL 계층에 구현된 실제 GPS 라이브러리를 동적으로 로딩한다. 'GPS HAL 라이브러리'는 GPS 장치 메이커가 제공한 것이거나 에뮬레이터용으로, 본 고는 에뮬레이터용 라이브러리를 기준으로 한다. 어느 라이브러리나 동일한 인터페이스를 제공한다. '라이브러리 로딩'이 성공하면 라이브러리에서 GPS 모듈 정보를 검색한다. '모듈 검색'이 성공한 후 모듈 정보를 통해 장치 열기를 시도한다. '장치 열기'가 성공하면 GPS 인터페이스 가져오기를 시도한다. 'GPS 인터페이스 가져오기'의 성공으로 GPS 지원 여부가 확인된다. 단, 여기서는 GPS 인터페이스 가져오기가 성공하는지 여부만 확인하며, 가져 온 인터페이스를 저장하지는 않는다. GPS 인터페이스는 GPS 장치의 초기화(init), 가동(start), 정지(stop) 등 GPS 장치를 제어할 수 있는 HAL 계층 함수들로 구성된다.

GPS 인터페이스 지원이 확인되면 본격적인 GPS 위치 제공자 로딩이 시작된다. 그림 3과 같이, 먼저 GLP 클래스 객체가 생성된다. 또한, GLP 객체의 생성 말미에 GLP 쓰레드가 생성 실행된다. 마지막으로, 생성된 GLP 객체가 위치 제공자 목록에 추가된다.

생성된 GPS 위치 제공자는 다른 위치 제공자들과 함께 순차적으로 초기화된다. LMS 쓰레드는 이를 위해 GLP 쓰레드에게 ENABLE 메시지를 송신함으로써 GPS 위치 제공자의 초기화를 시작한다. GPS 위치 제공자의 실제 초기화는 GLP 쓰레드에 의해 실행된다. 이로써 GPS 위치 제공자의 로딩이 완료된다.

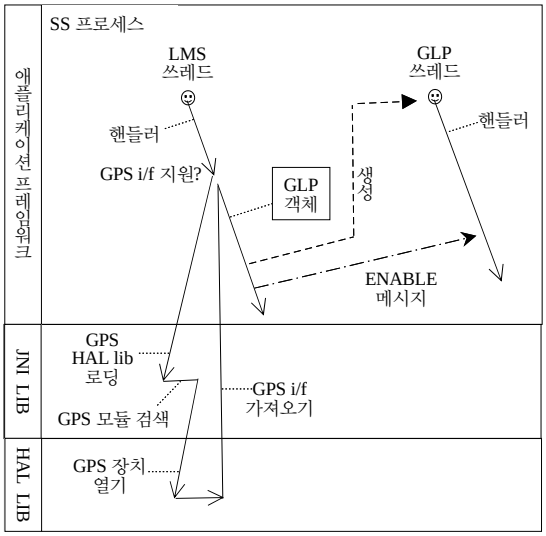

그림 3. LMS 쓰레드의 GPS 위치 제공자 로딩
Fig. 3 Loading GPS location provider by LMS thread

2.4. GLP 쓰레드와 GPS 위치 제공자 초기화

GLP 쓰레드는 GPS 위치 제공자의 초기화와 가동, 정지 등을 담당한다. GLP 쓰레드는 LMS 쓰레드에 의해 생성되어 실행될 때 그림 3과 같이, 메시지 핸들러를 준비한다. GLP 쓰레드는 LMS 쓰레드가 송신한 메시지를 메시지 핸들러를 통해 수신한다.

GPS 위치 제공자 초기화는 LMS 쓰레드로부터 ENABLE 메시지를 수신함으로써 시작된다. 메시지를 수신한 GLP 쓰레드는 그림 4와 같이, JNI 네이티브 함수를 호출하여 초기화를 실행한다. 이 JNI 초기화 함수는, 2.3절에서 언급된 GPS 인터페이스 지원 여부 확인 과정과 동일한 과정을 거쳐, GPS 인터페이스를 가져와 저장한다.

획득된 GPS 인터페이스에는, HAL 계층에 동적 로딩 라이브러리로 구현된 GPS 위치 제공자 제어 함수들

이 포함되어 있다. 이 함수들 중 init 함수가 실제 초기화를 수행한다.

따라서, GLP 쓰레드는, 콜백 함수 포인터를 인수로 하여, 그림 4와 같이, HAL 계층의 init 함수를 호출한다. 콜백 함수 포인터에는 JNI 함수 중 나중에 HAL 계층에서 역으로 호출되는 함수들이 포함된다. 예로, GPS 장치에서 위치 정보가 수신되었을 때 이의 보고를 위해 호출되는 '위치 정보 보고' 함수, 또는, '자바 쓰레드 생성' 함수를 들 수 있다.

2.4.1. HAL 계층의 초기화

호출된 init 함수는 그림 4와 같이, GPS 위치 제공자의 실제 초기화를 수행한다. 먼저, 하나의 파이프로 구현된 GPS 에뮬레이션 장치를 열어 파일 디스크립터(fd)를 얻는다. 하드웨어 GPS 장치의 경우, 하나의 시리얼 장치로 리눅스 커널에 포함되므로 역시 장치를 열어 fd를 얻는다.

다음, gps_state_thread (GST) 쓰레드를 생성한다. 이 쓰레드는 GPS 장치를 모니터링 하다가 위치 정보를 수신하여 이를 상위계층에 전달하는 쓰레드이다. 이 쓰레드는 콜백 함수 중 '자바 쓰레드 생성' JNI 함수가 호출되어 생성된다.

GLP 쓰레드가 GST 쓰레드를 제어하기 위한 통로로 이용할 소켓 쌍을 이용한다. 즉, GLP 쓰레드는 소켓 쌍의 한 쪽 끝에서 명령을 송신하며, GST 쓰레드는 다른 쪽 끝에서 명령을 수신한다.

이상, 초기화의 주요 결과인 GPS 장치 fd와 한 쌍의 소켓 디스크립터 값 및 넘겨받은 콜백 함수 포인터들은 Gps State (GS)변수에 저장되고, 두 쓰레드에 의해 공유된다.

2.4.2. GST 쓰레드와 초기화

GST 쓰레드는 GPS 장치로부터 위치 정보를 수신하여 이를 위도, 경도 및 고도 등으로 변환하여, 상위계층에 보고한다.

이를 위해, GST 쓰레드는 그림 4와 같이 , GLP 쓰레드에 의해 생성될 때 Nmea Reader (NR) 변수를 생성한다. 이 변수는 GPS 장치로부터 위치 정보를 수신하여 저장하기 위한 버퍼와, 변환된 위치 정보를 담는 Gps Location (GL) 변수를 멤버로 한다. 또한, 위치 정보 보고를 위한 콜백 함수 포인터를 담는 변수도 멤버로 갖는데, 이 콜백 변수는 다음 3장에서와 같이 GLP 쓰레드의 가동 명령에 의해 '위치 정보 보고' 함수로 설정되며, 지금은 NULL로 초기화 된다.

NR 변수를 만들고 난 후, GST 쓰레드는 무한 루프롤 돌며 GPS 장치 fd와, 소켓 쌍의 한 쪽 끝에서 각각 데이터 또는 명령이 수신되기를 기다린다.

이상으로, GPS 위치 제공자의 초기화 과정이 모두 완료된다.

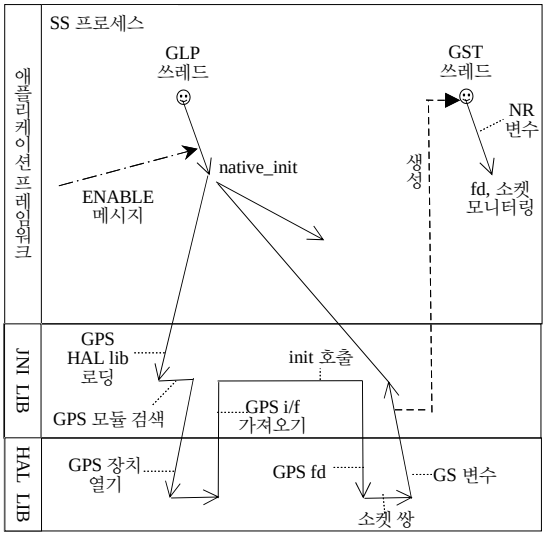

그림 4. GLP 쓰레드의 GPS 위치 제공자 초기화
Fig. 4 Initializing GPS location provider by GLP thread

Ⅲ. GPS 위치 제공자의 가동

이 장은 2 장과 같이 초기화된 GPS 위치 제공자의 가동(start) 과정을 분석한다. GPS 위치 제공자의 가동은 전형적으로 클라이언트 응용 프로그램의 위치 정보 요청에서 시작된다.

3.1. 응용 프로그램의 위치 정보 요청

위치 정보를 필요로 하는 클라이언트 응용 프로그램은, 먼저 2.1절에서 생성된 LMS 서비스에 연결되어야 한다[4]. 이는 시스템 서비스 획득 메서드를 호출하여 이루어지며, 이 메서드는 LMS 서비스에 연결되어 프록시 역할을 하는 Location Manager (LM) 객체를 반환한다.

이어서, 클라이언트는 위치 정보를 수신 받기 위한 '위치 정보 갱신' 콜백 메서드가 재정의된 Location Listener (LL) 객체, 또는 Pending Intent (PI)를 생성하고, 이를 LM 객체를 이용하여 그림 5와 같이, LMS 서비스 객체에 GPS 위치 제공자를 지정하여 등록한다. LMS 객체는 등록된 LL 객체, 또는 PI를 Receiver 객체로 wrapping 하고, 나중에 검색을 위해 GPS 위치 제공자의 records 목록에 추가한다.

이와 같이, 클라이언트의 위치 요청이 등록되면, LMS 객체는 GLP 쓰레드에게 ENABLE_TRACKING 메시지를 송신함으로써 GPS 위치 제공자의 가동 과정을 시작시킨다.

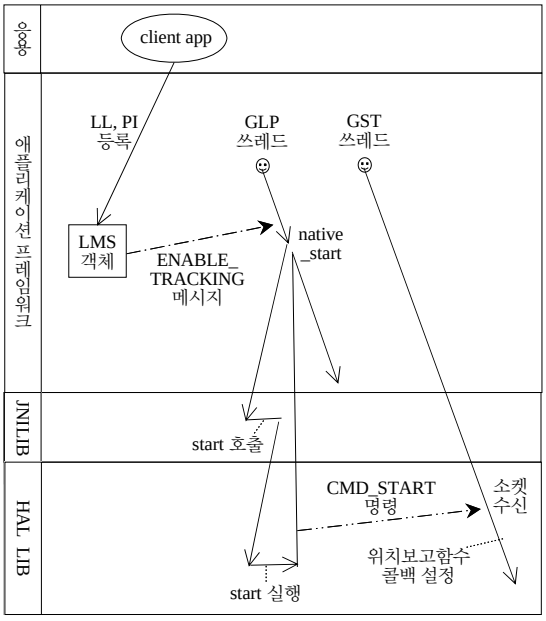

그림 5. GPS 위치 제공자의 가동
Fig. 5 Start of GPS location provider

3.2. GLP 쓰레드의 GPS 위치 제공자 가동

ENABLE_TRACKING 메시지를 수신한 GLP 쓰레드는 그림 5와 같이, GPS 위치 제공자의 가동(start)을 위해, JNI 함수를 거쳐 HAL 계층 GPS 인터페이스의 start 함수를 호출한다. start 함수는 소켓 쌍을 통해 GST 쓰레드에게 CMD_START 명령을 송신한다.

3.3. GST 쓰레드의 GPS 위치 제공자 가동

2.4.1절에 기술한 바와 같이 위치 제공자 초기화 과정의 하나로 생성된 GST 쓰레드는, 소켓을 모니터링하다가, GLP 쓰레드로부터 CMD_START 명령을 수신하면, 그림 5와 같이, NR 변수의 콜백 함수 포인터를 '위치 정보 보고' 함수를 가리키도록 설정한다. 이로써, GST 쓰레드는 앞으로의 위치 정보 보고를 이 함수를 호출하여 수행할 수 있다. 이로써, GPS 위치 제공자의 가동이 완료된다.

Ⅳ. GPS 위치 제공자의 위치 정보 보고

이 장에서는, 앞 두 장에서 살펴 본 바와 같이 초기화 과정과 가동을 마친 GPS 위치 제공자를 통해, GPS 위치 정보가 보고되는 과정을 살펴본다.

4.1. GST 쓰레드의 위치 정보 수신 및 처리

GST 쓰레드는, 파이프로 구현된 에뮬레이터 GPS 장치 fd나, 시리얼 장치로 커널에 적재된 하드웨어 GPS 장치 fd로부터, 문자열로 된 위치 정보를 수신한다. 그림 6 참조. GST 쓰레드는 이 문자열을 NR 변수의 버퍼에 저장한 후, 위치 정보 변환(parsing)을 수행한다. 즉, 문자열로 구성된 위치 정보를 위도, 경도, 고도 등의 위치 정보로 변환하고, 이를 NR 변수의 GL 변수에 저장한다.

변환이 완료되면 3.3절에서 NR 변수에 설정된 콜백 함수를 호출하여 그림 6과 같이, 상위 계층으로 위치 정보 보고를 시작한다. 호출된 콜백 함수는 JNI 함수를 통해 자바 계층의 GLP 객체와 LMS 객체의 위치 보고 메서드로 연결된다. LMS 객체는 LMS 쓰레드에게 MESSAGE_LOCATION_CHANGED (MLC) 메시지를 송신한다.

4.2. LMS 쓰레드의 위치 정보 보고

MLC 메시지를 통해 GPS 위치 정보를 수신한 LMS 쓰레드는, 3.1절에서 등록된 GPS 위치 제공자의 모든 records를 검색하여 Receiver 객체를 찾고, 이를 통해 LL 객체 또는 PI를 추출한다. 수신된 위치 정보는 LL 객체의 경우, '위치 정보 갱신' 콜백 메서드를 호출하여 클라이언트에게 최종적으로 보고한다. PI의 경우는 위치 정보를 인텐트에 실어 클라이언트에게 전송한다. 이상으로, GPS 위치 제공자를 통해 GPS 장치로부터 클라이언트까지 위치 정보 전달이 최종 완료된다.

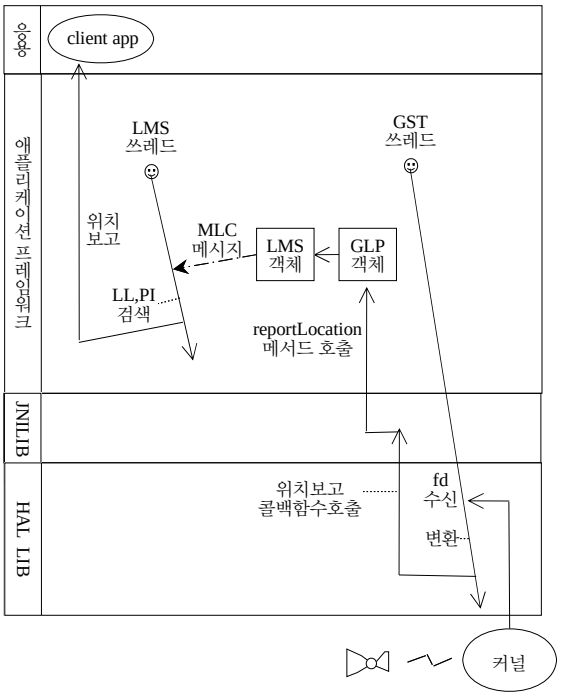

그림 6. GPS 위치 정보 보고
Fig. 6 Report of GPS location

Ⅴ. 관련 소스 파일

표 1은 본 고에서 참조한 소스 파일들을 요약 정리한다. 표에서 보는 바와 같이 관련 소스 파일들을 계층별로 분류하여 정리하였다. 구글 안드로이드 소스 [3]중 최신 소스라 할 수 있는 tag 4.1.2_r2.1을 기준으로 하였다. 다만, GPS 인터페이스가 구현된 gps_qemu.c 파일은 tag 4.1.2_r2.1에 포함되지 않아, tag 4.3_r1을 참고하였다. 안드로이드 소스는 Git Hub 사이트[5]에서도 검색할 수 있다.

Ⅵ. 결 론

안드로이드 프레임워크의 한 서비스로 구현된 GPS 위치 제공자에 대한 상당한 양의 소스를 읽고 동작을 이해하는 일은 시간이 오래 걸리고 어려운 일이다. 하지만, 본 고를 통해 그 고통을 대폭 경감할 수 있으리라 기대한다. 먼저, 본 논문을 통해 위치 제공자를 구성하는 주요 객체와 쓰레드의 기능을 단계별로 이해한 후, 참고문헌들[1,2]을 통해 소스 레벨에서 함수 호

표 1. 관련 소스 파일의 경로 및 내용
Table. 1 Paths and contents of related source files

계층	소스경로(https://android.googlesource.com/)	소스 파일명	관련 내용
애플리케이션 프레임워크	platform/frameworks/base/+/android-4.1.2_r2.1/services/java/com/android/server/	SystemServer.java	SS 프로세스, JNI lib 로딩, 다양한 시스템 서비스 시작, Server 쓰레드, LMS 서비스 시작
		LocationManagerService.java	LMS 서비스 구현, LMS 객체, LMS 쓰레드, LMS 핸들러, 위치 제공자 로딩
	platform/frameworks/base/+/android-4.1.2_r2.1/services/java/com/android/server/location/	GpsLocationProvider.java	GPS 위치 제공자 구현, GLP 객체, GLP 쓰레드, GLP 핸들러, reportLocation 메서드
JNI 계층	platform/frameworks/base/+/android-4.1.2_r2.1/services/jni/	com_android_server_location_GpsLocationProvider.cpp	GPS 서비스를 위한 JNI 네이티브 함수, 콜백함수, 네이티브 메서드와 함수 매핑
HAL 라이브러리	device/generic/goldfish/+/android-4.3_r1/gps/	gps_qemu.c	HAL 계층 GPS 라이브러리, GPS 인터페이스의 구현(init, start, stop 등), GST 쓰레드
	platform/hardware/libhardware/+/android-4.1.2_r2.1/include/hardware/	gps.h	다양한 GPS 장치를 위한 공통 인터페이스, GpsInterface, GpsCallbacks, GpsLocation 정의
		hardware.h	하드웨어 module, methods, device 자료구조 정의
	platform/hardware/libhardware/+/android-4.1.2_r2.1/	hardware.c	HAL 계층의 GPS 라이브러리 검색 및 동적 로딩

출 예시를 살펴 본 후, 마지막으로 소스 자체를 분석해 볼 것을 추천한다. 물론, 이를 위해서는 가장 먼저 안드로이드 프레임워크의 구조와 JNI 같은 관련 배경 지식[4]과, 안드로이드 쓰레드간 메시지 통신 지식[6]이 필요하다.

감사의 글

이 논문은 2012학년도 동의대학교 교내연구비에 의해 연구되었음(2012AA170)

REFERENCES

[1] H. C. Ko, H. M. Yu, “Android GPS Subsystem,” in *Android - everything and porting*, Hanbit media, ch. 6, pp 201-238, 2011.

[2] K. Lee, “An analysis on the initialization of the GPS Location Provider in the Android,” *Journal of Research Institute of Industrial Technology Development*, vol. 27, pp. 157-161, Jan. 2013.

[3] The Android Open Source Project. Available: https://android.googlesource.com.

[4] H. J. Song and et. al., *Inside the Android Framework*, Wiki Books, 2010.

[5] Git Hub for the Android Open Source Project. Available: www.github.com/android.

[6] S. H. Kim, “Thread,” in *Android Programming Complete Guide* vol. 2, Hanbit media, ch. 16, pp 842-901, 2011.

이계상(Kyesang Lee)

KAIST 전기및전자공학과 공학박사
현 동의대학교 정보통신공학과 교수
※관심분야 : 차세대 인터넷 프로토콜, 안드로이드 플랫폼 등.