**Android Wi-Fi工作原理**

刘 洋

# 第一部分 Android概述

在介绍Wi-Fi之前，先简要介绍一下Android系统，主要分析一下Android的按层实现的原理。Android层次结构是整个Android体系中所有应用实现的基础框架，而Android源代码结构则与Wi-Fi的实现细节有关。

## 1.1 基础知识

Android是一款当前最为流行的手机操作系统，它本身的开放性加上Google公司的大力推广，使其获得了大量手机生产厂商、科研院校、软件公司以及个人开发者的青睐，它属于一个全开放的平台，因此开发者可以得到整个系统的源代码，并能对其进行修改，修改的结果可以通过互联网上传到Android官方网站，倘若被审核通过，就能加入到Android的源代码中，这绝对是一件令人兴奋的事情。

## 1.2 Android层次结构

Android系统是在Linux系统的基础上，经过了层层封装，最终提供给开发者的是大量的Java API，在这里被叫做Android API，于是，开发者就可以像开发一般的Java程序那样开发Android应用程序，这样的设计不仅降低了开发Android应用程序的难度，还增加了Android系统的界面友好度。

和一般的操作系统一样，Android也是对硬件进行了多层的封装，使得应用程序的开发者和用户能轻松地操作硬件，完成他们所希望完成的事情。Android所针对的硬件就是手机，这里主要指智能手机，这种智能手机与传统的手机相比电话功能被弱化，而更偏向于一台笔记本电脑，因此它的CPU、内存等硬件配置要比传统的手机高。它需要提供给用户一些电脑所拥有的功能，比如说Wi-Fi上网、鼠标或触屏控制的界面、收发电子邮件、玩大型游戏等，但同时又必须拥有传统手机所支持的电话、摄像头、蓝牙等功，这些挑战都增加了Android的设计难度。

Android从下至上可以分为这样几个层次：

（1）Linux内核及驱动层（C实现）；

（2）本地库（C库和C++库）和Java运行时环境层（主要由C、C++实现）；

（3）Java框架层（主要由Java实现）；

（4）Java应用程序层（Java实现）。

第3层和第4层之间就是上文所说的Android API，这也是Android提供给应用程序开发人员的接口，我们只要熟悉了这些API，就可以进行Android应用程序的开发工作了。

Android手机中所有的额应用程序，包括核心应用程序和用户开发的应用程序，它们都属于第4层次，用户可以得到系统自带的所有程序的源代码，比如初始界面管理程序、短信程序、日历、联系人管理程序等，并能随意修改这些程序，甚至还可以删除其中的一两个，然后重新编译源代码，这样便生成了自己定制的Android系统。

## 1.3 Android源代码结构

虽然只看API就可以编写Android的应用程序了，但是无论对Android系统的研究人员还是应用程序的开发者来说，Android的源代码都是一笔值得好好研究的财富。

Android的源代码可以从其官方网站上下载到，在此之前，本机上需要装Linux系统，然后根据官网上的步骤逐步执行，包括初始化系统环境、下载源码、编译源码，就可以在本机上下载并编译整个Android系统的源代码。讲述如何下载并编译Android源代码的官网地址如下：<http://source.android.com/source/initializing.html>。

纵观整个Android源码的结构，在其根目录下有大约10多个文件夹，其中可以大致分为以下三部分：

1. 关键部分

这部分的实现代码位于根目录下除了external和package之外的所有文件夹中，这些代码实现了Linux内核 (kernel文件夹)、核心驱动、Android驱动、Android系统的建立 (build文件夹)、Dalvik虚拟机 (dalvik文件夹)、C和C++本地库 (bionic文件夹)、硬件抽象、无线硬件接口 (hardware/ril文件夹)、Java运行环境的支持等功能，是整个Android系统的启动和运行所必须的。

1. 扩展部分

这部分的实现代码位于external文件夹中，这里边存放着许多其他开源项目，这些项目都已经过修改而融入到Android系统中。

1. 应用程序包

这部分代码位于package文件夹中，由应用程序（apps）、提供器（providers）、输入法（input methods）三部分组成。

Android手机自带的应用程序就位于./package/apps中，这里有Browser（浏览器）, AlarmClock（闹钟）, Camera（照相机）, Contacts（联系人）, Settings（设置）, Launcher（初始界面）等，还有一些，在此不再一一罗列。

# 第二部分 Wi-Fi层次结构

Android中Wi-Fi驱动程序被编译成内核的模块，通过应用程序设置开关进行加载和卸载，具体来说就是Settings --> Wireless & networks --> Wi-Fi。同时，要使Wi-Fi正常工作，驱动中还需要实现烧写固件程序和配置信息到Wi-Fi的芯片中。

## 2.1 Wi-Fi程序模块

在Android的源代码中，有多处地方涉及到Wi-Fi，跨越了前文中所说的1、2、3、4层。下面介绍几个与实现Wi-Fi功能密切相关的程序模块。

### 2.1.1 开源库wpa\_supplicant

它是一个开源的库，加入到Android源码中，经过修改后成为Android实现Wi-Fi功能的基础。它的代码位于./external/wpa\_supplicant文件夹中，主要用C和C++写成，实现了从上层接到命令后，发送给硬件驱动程序，接着操作硬件完成需要的操作，这里是通过socket来与硬件驱动进行通信的。下图2-1是wpa\_supplicant的框架图。

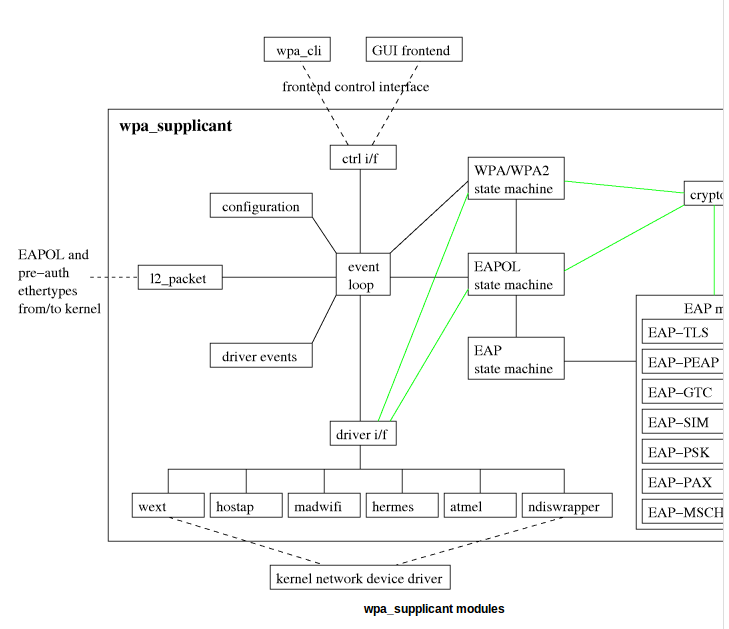


图2-1 wpa\_supplicant开源项目框架图

### 2.1.2 硬件驱动程序

前文所说的wpa\_supplicant与之通信的硬件驱动的代码位于./hardware/libhardware\_legacy/wifi/wifi.c中。

### 2.1.3 JNI部分

首先简要介绍一下JNI，JNI是Java Native Interface的缩写，它实现了Java代码与其他代码进行交互，使得在Java虚拟机中运行的Java代码能够与用其他语言编写的应用程序和库进行交互。在Android中，JNI可以让Java程序调用C程序。

与Wi-Fi相关的JNI代码位于./frameworks/base/core/jni/android\_net\_wifi\_Wifi.cpp中。

### 2.1.4 Wi-Fi API部分

这部分源代码使用Java完成了对Wi-Fi API的封装，使应用程序可以使用Wi-Fi功能，它们位于frameworks/base/services/java/com/android/server/和frameworks/base/wifi/java/android/net/wifi/中。

### 2.1.5 Wi-Fi Settings应用程序部分

这是Android中自带的一个应用程序，在手机的Settings中，它可以让用户手动打开或关闭Wi-Fi功能。当用户打开Wi-Fi功能后，它会自动搜索周围的无线网络，并以列表的形式显示，供用户选择，默认会连接用户上一次成功连接的无线网络。这部分代码位于./packages/apps/Settings/src/com/android/settings/wifi中。

## 2.2 Wi-Fi层次结构关系

下图2-2就是Android中Wi-Fi的各模块在整个Android层次结构中的位置，以及它们之间的关系。



图2-2 Android中Wi-Fi的层次结构图

# 第三部分 Wi-Fi执行过程

Android中Wi-Fi是使用层次结构设计的，因此执行过程基本上是在接到用户命令后，先从上到下，再从下到上，完成用户与Wi-Fi设备的交互。下图3-1就是Wi-Fi功能的详细执行过程示意图。

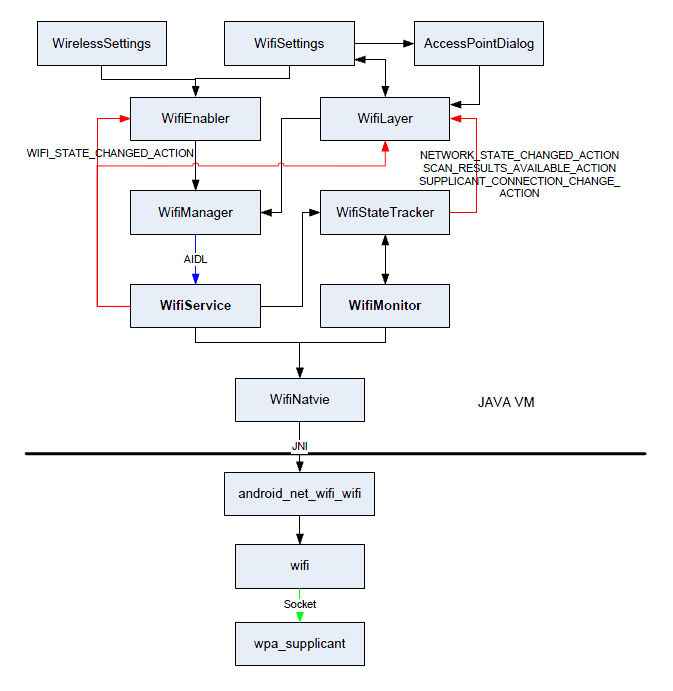


图3-1 Wi-Fi执行过程示意图

如上图3-1所示，Wi-Fi的执行过程主要有4个，下文将对这几个过程进行详细介绍。

## 3.1 从Settings中启动Wi-Fi

当用户按下Wi-Fi 按钮后，Android 会调用WifiEnabler 的onPreferenceChange，再由WifiEnabler调用WifiManager 的setWifiEnabled 接口函数，通过AIDL，实际调用的是WifiService 的setWifiEnabled 函数，WifiService 接着向自身发送一条MESSAGE\_ENABLE\_WIFI 消息，在处理该消息的代码中做真正的使能工作：首先装载WIFI 内核模块（该模块的位置硬编码为"/system/lib/modules/wlan.ko" ）， 然后启动wpa\_supplicant （ 配置文件硬编码为"/data/misc/wifi/wpa\_supplicant.conf"），再通过WifiStateTracker 来启动WifiMonitor 中的监视线程。

当使能成功后，会广播发送WIFI\_STATE\_CHANGED\_ACTION 这个Intent 通知外界Wi-Fi已经成功使能了。WifiEnabler 创建的时候就会向Android 注册接收

WIFI\_STATE\_CHANGED\_ACTION，因此它会收到该Intent，从而开始扫描。

## 3.2 查找Access Point (AP)

扫描的入口函数是WifiService 的startScan，它其实也就是往wpa\_supplicant 发送SCAN 命令。当wpa\_supplicant 处理完SCAN 命令后，它会向控制通道发送事件通知扫描完成，从而wifi\_wait\_for\_event 函数会接收到该事件，由此WifiMonitor 中的MonitorThread 会被执行来出来这个事件，WifiStateTracker 则接着广播发SCAN\_RESULTS\_AVAILABLE\_ACTION 这个Intent ，WifiLayer 注册了接收SCAN\_RESULTS\_AVAILABLE\_ACTION 这个Intent，所以它的相关处理函数handleScanResultsAvailable 会被调用，在该函数中，先会去拿到SCAN 的结果（最终是往wpa\_supplicant 发送SCAN\_RESULT 命令并读取返回值来实现的）。

对每一个扫描返回的AP，WifiLayer 会调用WifiSettings 的onAccessPointSetChanged 函数，从而最终把该AP 加到GUI 显示列表中。

## 3.3 连接AP

当用户在 AcessPointDialog 中选择好加密方式和输入密钥之后，再点击连接按钮，Android就会去连接这个AP。

WifiLayer 会先检测这个AP 是不是之前被配置过，这个是通过向wpa\_supplicant 发送LIST\_NETWORK 命令并且比较返回值来实现的，如果wpa\_supplicant 没有这个AP 的配置信息，则会向wpa\_supplicant 发送ADD\_NETWORK命令来添加该AP，ADD\_NETWORK 命令会返回一个ID ，WifiLayer 再用这个返回的ID 作为参数向wpa\_supplicant 发送ENABLE\_NETWORK 命令，从而让wpa\_supplicant 去连接该AP。

## 3.4 配置IP地址

当 wpa\_supplicant 成功连接上AP 之后，它会向控制通道发送事件通知连接上AP 了，从而wifi\_wait\_for\_event 函数会接收到该事件，由此WifiMonitor 中的MonitorThread 会被执行来处理这个事件，WifiMonitor 再调用WifiStateTracker 的notifyStateChange，WifiStateTracker 则接着会往自身发送EVENT\_DHCP\_START 消息来启动DHCP 去获取IP 地址，然后再广播发送NETWORK\_STATE\_CHANGED\_ACTION 这个Intent。

WifiLayer 注册了接收NETWORK\_STATE\_CHANGED\_ACTION 这个Intent，所以它的相关处理函数handleNetworkStateChanged 会被调用，当 DHCP 拿到IP 地址之后，会再发送EVENT\_DHCP\_SUCCEEDED 消息，WifiLayer 处理EVENT\_DHCP\_SUCCEEDED 消息， 会再次广播发送NETWORK\_STATE\_CHANGED\_ACTION 这个Intent，这次带上完整的IP 地址信息。至此为止，整个连接过程完成。