文章编号：1671-1742(2010)05-0495-05

Android 系统中 Wi-Fi 网络的研究与实现

陈法海， 杨 斌
（西南交通大学信息科学与技术学院，四川 成都 610031）

摘要：对时下流行的 Android 系统中 Wi-Fi 网络的实现技术进行了研究。在详细分析 Wi-Fi 模块的系统组成的基础上，从系统使用及编程者的角度，深入剖析了 Wi-Fi 模块的初始化、启动、接入点扫描和 IP 地址配置的实现，通过移植及调试 Wi-Fi 驱动程序，成功验证了 Wi-Fi 网络的实现过程。

关 键 词：计算机应用技术；嵌入式系统；Android；Wi-Fi 模块；接入点
中图分类号：TP316 **文献标识码**：A

1 Android 系统概述

Android[1]是 Google 公司开发的基于 Linux 平台开源的手机操作系统，该平台由操作系统、中间件、用户界面和应用软件组成，具体框架结构由 5 部分组成，其核心为 Linux2.6 内核，向上搭配 Libraries（函数库）及 AndroidRuntime（运行环境），再配合 Application Framework（应用程序框架），来开发各种不同的 Application（应用程序），是首个为移动终端打造的开放和完整的移动软件[2]。

Android 系统基于 Linux 2.6 内核来提供系统的核心服务，例如安全机制，内存管理，进程管理，网络堆栈和驱动模块。其包含一组核心库，提供了 Java 语言核心库内的大部分功能。Android 应用程序运行于 Dalvik 虚拟机上，Dalvik 虚拟机是基于寄存器的，编译器将 Java 源文件转为 class 文件，内置的 dx 工具又将 class 文件转化为 Dex 文件，Dex 文件是在 Dalvik 虚拟机上运行程序的标准格式[3]。在 Wi-Fi 网络方面，虽然 Android 建立在 Linux 内核之上，但是 Wi-Fi 网络的实现与一般 Linux 操作系统不一样，下面结合 Android2.1 的源码，对 Wi-Fi 模块的工作原理进行了分析。

2 Wi-Fi 模块的工作原理

2.1 Wi-Fi 模块的组成

在 Android 系统中，应用程序可以使用 Wifimanager 提供的 API 接口管理 Wi-Fi 的连接及使用情况，比如：启动或者禁止 Wi-Fi 网络、请求扫描接入点（Access Point，AP）、返回网络配置参数列表等等。当 Wi-Fi 网络被启动或者禁止时，Wifiservice 将能过广播方式发出 WIFI_STATE_CHANGED_ACTION 事件通知上层应用程序，Wifiservice 主要负责对 WifiMonitor 和 wpa_supplicant 的进程进行初始化及禁止，并且发出命令给 wpa_supplicant。WifiMonitor 是负责接收 wpa_supplicant 发出的各种事件通知，然后 WifiStateTracker 广播各种行为。Wi-Fi 模块的组成如图 1 所示。

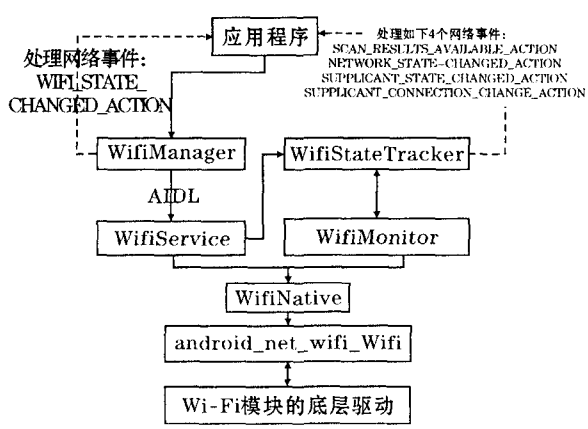

图 1 Wi-Fi 模块的系统组成

在 Wi-Fi 模块的系统组成中 Wifimanager 主要提供一些 API 接口，Wifimanager 处理及工作的内容大致如下几个方面：

(1)返回网络配置的参数列表，参数列表可以查看及更新，并且个别的配置参数可以被修改。

(2)建立连接网络及禁止，以及查询有关网络状态的动态信息。

(3)扫描无线网络 AP，并且根据 AP 的信息进行连接。

收稿日期：2010-10-22

(4)定义各种 Intent 组件的行为名称,组件行为是广播来更新 wifi 状态。

Wi-Fi 模块中的 Wifiservice 可以通过 WifiManager 接口处理远程无线 Wi-Fi 操作请求,也可以创建一个 WifiMonitor 来侦听无线上网相关的事件。WifiStateTracker 可以跟踪 Wi-Fi 连接状态。有关的无线网络事件的处理及状态更新都可以通过 WifiStateTracker 完成。WifiMonitor 侦听从 wpa_supplicant 服务器发出的事件请求,并且处理事件请求然后发送到 WifiStateTracker,WifiMonitor 独立运行于自己的线程中。WifiNative 要求发送请求守护进程。WifiStateTracker 所处理的事件如下:

(1) WIFI_STATE_CHANGED_ACTION:表示 Wi-Fi 网络已经被启动、禁止、正在启动中。正在禁止中和未知状态中。

(2) SUPPLICANT_STATE_CHANGED_ACTION:表示连接一个 AP 的状态信息已经改变了,并且系统提供一个新状态信息。

(3) SUPPLICANT_CONNECTION_CHANGE_ACTION:表示与请求者的连接已经建立或者已经断开。

(4) SCAN_RESULTS_AVAILABLE_ACTION:表示 AP 扫描已经完成,并且返回扫描信息。

(5) RSSI_CHANGED_ACTION:表示 Wi-Fi 的信号强度已经改变。

(6) NETWORK_STATE_CHANGED_ACTION:表示 Wi-Fi 网络的连接状态已经改变。

(7) NETWORK_IDS_CHANGED_ACTION:表示配置网络的网络 ID 号已经改变。

Android 实现 Wi-Fi 网络大致经过 4 个步骤:Wi-Fi 模块的初始化;Wi-Fi 模块的启动;AP 扫描及配置 AP 参数;Wi-Fi 连接及配置 IP 地址。下面是 Android2.1 中实现 Wi-Fi 上网的 4 个步骤的源码分析。

2.2 Wi-Fi 模块的初始化

当 Android 系统启动 Wi-Fi 模块时 ,先对 Wi-Fi 模块进行初始化,Android 系统通过 startService(Intent service)可以启动一个 Service,通过 Context.bindService()可以绑定一个 Serivce。在系统启动 SystemServer 的时候,通过 ServiceManager 调用 addService()函数生成一个 ConnectivityService 实例。在 ConnectivityService 的构造函数里面创建 WifiService 和 WifiStateTracker,而 WifiStateTracker 却可以创建 WifiMonitor 接收来自底层的事件,WifiService 和 WifiMonitor 是整个 Wi-Fi 模块的核心。WifiService 负责启动关闭 wpa_supplicant、启动关闭 WifiMonitor 监视线程和把命令下发给 wpa_supplicant,而 WifiMonitor 则负责从 wpa_supplicant 接收事件通知。Wi-Fi 模块初始化过程如图 2 所示。

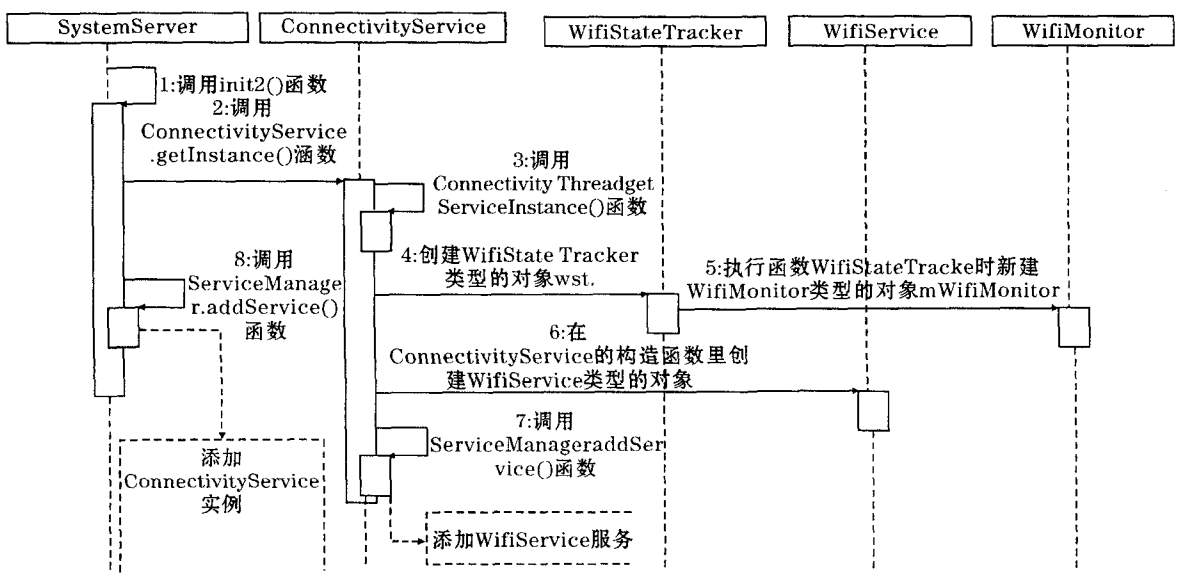

图 2 Wi-Fi 模块初始化流程图

2.3 Wi-Fi 模块的启动

Android 系统中 WirelessSettings 应用程序是负责启动 Wi-Fi 模块的,会调用 WifiEnabler 函数处理 Wi-Fi 按钮。当用户按下 WiFi 按钮后,WirelessSettings 应用程序会调用 WifiEnabler 的 onPreferenceChange,再由

WifiEnabler 函数调用 WifiManager 的 setWifiEnabled 接口函数，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 Wi-Fi 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 启动 WifiMonitor 中的监视线程。当使能成功后,会广播发送 WIFI_STATE_CHANGED_ACTION 消息通知外界 Wi-Fi 已经成功启动。WifiEnabler 创建的时候就会向 Android 系统注册接收 WIFI_STATE_CHANGED_ACTION,因此当接收到该 Intent,系统就开始扫描 AP。Wi-Fi 模块启动过程如图 3 所示。

2.4 AP 扫描及配置 AP 参数

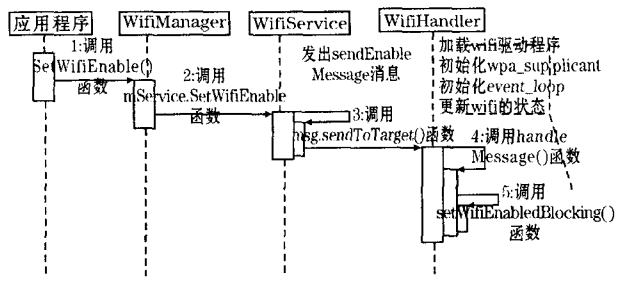

图 3　Wi-Fi 模块启动流程图

当 Android 扫描 AP 时,Wi-Fi 模块执行 AP 扫描的函数 startScan(),实质是给 wpa_supplicant 发送 SCAN 命令。当 wpa_supplicant 接收处理 SCAN 命令后,wpa_supplicant 会向控制通道发送事件通知扫描结束,当 wifi_wait_for_event 函数接收到该事件后,WifiMonitor 中的相应函数被调用处理该事件。同时 WifiStateTracker 接着广播发送 SCAN_RESULTS_AVAILABLE_ACTION,在 WifiLayer 注册了接收 SCAN_RESULTS_AVAILABLE_ACTION,所以相关处理函数 handleScanResultsAvailable 会被调用,在该函数中,先得到 AP 扫描的结果(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。AP 扫描过程如图 4 所示。

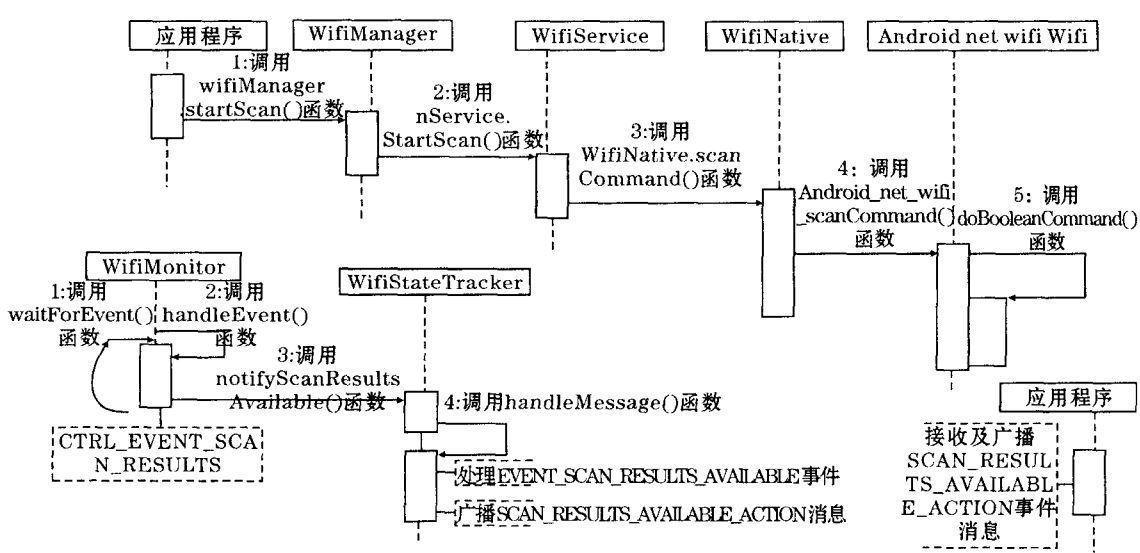

图 4　AP 扫描流程图

在 Android 系统的 WifiSettings 界面上选择了一个 AP 后,会显示配置 AP 参数的一个对话框,用户要在 AcessPointDialog 对话框中正确地选择 AP 参数。AP 参数配置过程如图 5 所示。

2.5 Wi-Fi 连接及配置 IP 地址

在 Android 系统的 AcessPointDialog 对话框中选择好加密方式和连接密钥之后,点击连接按钮,然后 Android 系统就会自动连接 AP。

WifiLayer 会通过向 wap_supplicant 发送 LIST_NETWORK 命令实现检测 AP 是否之前被配置过。如果 wpa_supplicant 没有 AP 的配置信息,就会向 wpa_supplicant 发送命令添加该 AP,WifiLayer 得到返回的 networkId,再利用 networkId 参数向 wpa_supplicantv 发送连接该 AP 命令,并且保存该 AP 配置信息为以后使用。

Wi-Fi 模块连接过程如图 6 所示。

当 wpa_supplicant 成功连接上 AP 之后，wpa_supplicant 会向控制通道发送事件通知已经连接上 AP，然后执行 WifiMonitor 中的 MonitorThread 处理该事件。

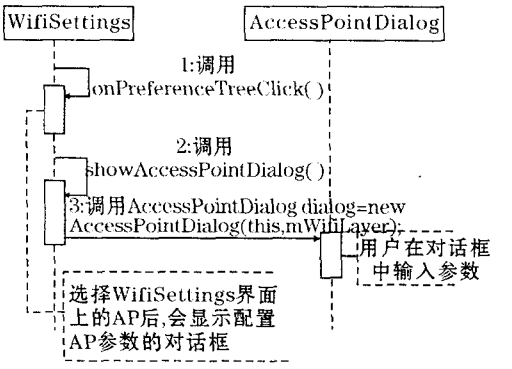

图 5 AP 参数配置流程图

Wi-Fi 模块连接到 AP 后，Android 系统就要配置 IP 地址，此时系统中的 WifiMonitor 再调用 WifiStateTracker 的 notifyStateChange 函数，接着 WifiStateTracker 会往自身发送 EVENT_NETWORK_STATE_CHANGED 消息启动 DHCP 去获取 IP 地址，当 DHCP 获取 IP 地址后，会发送 EVENT_INTERFACE_CONFIGURATION_SUCCEEDED 消息，当 WifiLayer 收到 EVENT_INTERFACE_CONFIGURATION_SUCCEEDED 消息后，会广播发送 EVENT_NETWORK_STATE_CHANGED，并且附带获取 IP 地址的完整信息，WifiLayer 中注册此 Intent 的接受者，并调用 handleNetworkStateChanged 函数处理消息。调用函数成功后 Android OS 的 IP 地址配置结束，Android 系统具有 Wi-Fi 上网功能。

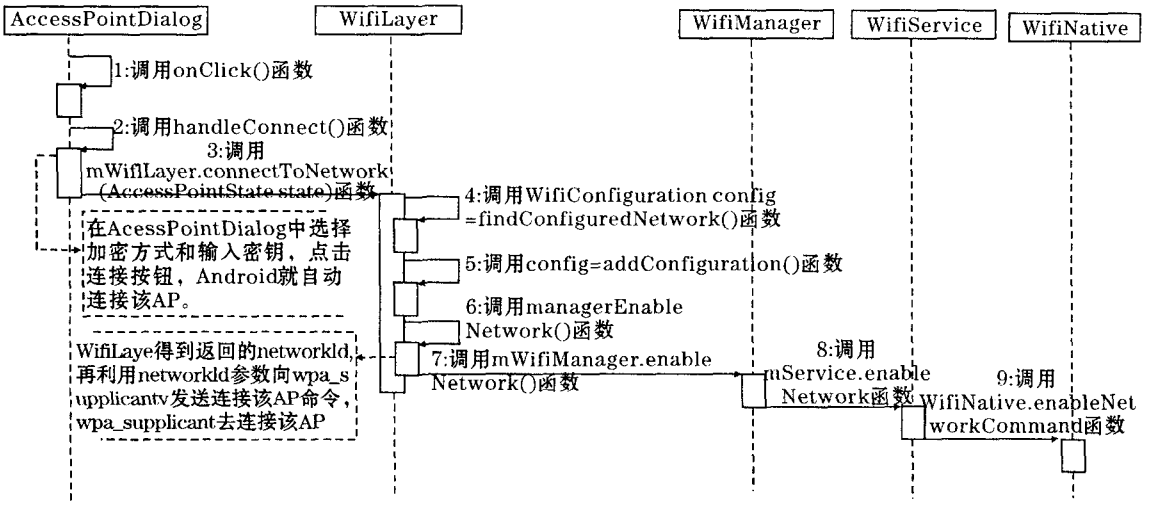

图 6 Wi-Fi 模块连接流程图

3 Wi-Fi 网络的实现

3.1 Wi-Fi 驱动的移植

Android 系统可以通过 Wi-Fi 模块无线上网，在 S3C6410 开发板中 Wi-Fi 模块与 S3C6410 处理器的接口是 SDIO，要让 Wi-Fi 模块正常工作，必须保证 SDIO 的驱动是配置正确，因此 Wi-Fi 模块的驱动移植需要配置两个地方，分别是配置 firmware 和 marvel8686SDIO 驱动。在内核配置界面中选择 firmware 的相关配置：Generic Driver Options->Prevent firmware from being built 和 Generic Driver Options->Userspace firmware loading support (NEW)->Include in-kernel firmware blobs in kernel binary，配置时要用到 2 个 marvel8686 的 firmware 文件。内核配置时选择 marvel8686SDIO 的相关配置：Network device support->Wireless LAN->Wireless LAN(IEEE 802.11)->Marvell Libertas WLAN driver support->Marvell Libertas 8385 and 8686 SDIO 802.11b/g cards、Network device support->Wireless LAN->Wireless LAN(IEEE 802.11)->Marvell Libertas WLAN driver support->Enable full debugging output in the Libertas module 和 Network device support->Wireless LAN->Wireless LAN(IEEE 802.11)->Marvell 8xxx Libertas WLAN driver support with thin firmware。退出内核配置界面后运行命令 make 来编译 Android 的内核。

3.2 Wi-Fi 模块的测试

通过 USB 将内核镜像文件下载到三星 S3C6410 开发板的内存中,下载结束后复位板子或者断电重启。在内核启动过程中,从超级终端显示 Wi-Fi 模块的信息:

mmc0: new SDIO card at address 0001
libertas_sdio mmc0:0001:1: firmware: using built-in firmware sd8686_helper.bin
libertas_sdio mmc0:0001:1: firmware: using built-in firmware sd8686.bin
libertas: 00:22:43:73:26:bf, fw 9.70.3p24, cap 0x000003a3
libertas: PREP_CMD: command 0x00a3 failed: 2
libertas: PREP_CMD: command 0x00a3 failed: 2
libertas: eth1: Marvell WLAN 802.11 adapter

以上显示的信息表明内核已经找到 WIFI 模块。

当 S3C6410 开发板成功运行 Android 系统后,选择 settings->wireless&network->WIFI,然后在超级终端中输入 logcat 命令可以显示 Wi-Fi 使用信息,并且开发板上 Wi-Fi 模块的指示灯 LED1 会闪烁。

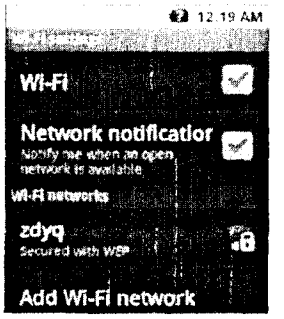

图 7　Android 搜索的 AP 结果

点击 Android 系统下的 Wifi Settings 对话框,Android 系统开始搜索 AP,搜索 AP 成功后界面显示出 AP 的情况,选择需要连接的 AP 进行连接。连接成功后 S3C6410 开发板具有了 Wi-Fi 网络功能,Android 系统中显示搜索的 AP 信息如图 7 所示。

4　结束语

对 Android 智能手机操作系统来说,Wi-Fi 网络系统是其中一个主要组成部分,了解 Android 系统中的 Wi-Fi 网络的工作原理可以对应用程序性能上的提供有所帮助。在 Android 系统移植到其他嵌入式设备中,Android 系统中 Wi-Fi 的底层驱动移植是其中一个关键部分,通过对底层 Wi-Fi 接口以及对 Wi-Fi 驱动移植的研究,将更有效地实现 Android 系统在其他嵌入式设备上的移植及开发相应 Wi-Fi 网络的应用程序。

参考文献:

[1] Android project official. Android project[EB/OL]. http://www.android.com/,2008.
[2] Code Home. Android-An Open Handset Alliance Project [EB /OL]. http://code.google.com / android /what is android. html, 2008.
[3] 陈憬,陈平华,李文亮. Android 内核分析[J]. 现代计算机(专业版). 2009,(11):112-114.
[4] 韩超,梁泉. Android 系统原理及开发要点详解[M]. 北京:电子工业出版社,2010.

Research and implementation of Wi-Fi network in Android system

CHEN Fa-hai,　YANG Bin
(School of Information Science & Technology, Southwest Jiaotong University, Chengdu 610031, China)

Abstract: The implementation technology of Android Wi-Fi was researched. Based on the detailed analysis of the composition of Android Wi-Fi module, the implementation of initialization, startting, AP scanning and IP address configuration of Wi-Fi module are deeply illustrated from both the system usage and programmer's aspects. By transplantation and debugging of the Wi-Fi drive program, the implementation progress of Wi-Fi network is validated successfully.

Key words: computer application; embedded system; Android; Wi-Fi module; access point