Wifi 网卡状态

1. WIFI_STATE_DISABLED: WIFI 网卡不可用
2. WIFI_STATE_DISABLING: WIFI 正在关闭
3. WIFI_STATE_ENABLED:WIFI 网卡可用
4. WIFI_STATE_ENABLING:WIFI 网卡正在打开
5. WIFI_STATE_UNKNOWN:未知网卡状态

WIFI 访问网络需要的权限

```
<uses-permission
android:name="android.permission.CHANGE_NETWORK_STATE">
</uses-permission>修改网络状态的权限
<uses-permission
android:name="android.permission.CHANGE_WIFI_STATE">
</uses-permission>修改WIFI状态的权限
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE">
</uses-permission>访问网络权限
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE">
</uses-permission>访问 WIFI 权限
```

WIFI 核心模块

- WifiService

由 SystemServer 启动的时候生成的 ConnecttivityService 创建，负责启动关闭 wpa_supplicant,启动和关闭 WifiMonitor 线程,把命令下发给 wpa_supplicant 以及跟新 WIFI 的状态

- WifiMonitor

负责从 wpa_supplicant 接收事件通知

- Wpa_supplicant

  1、读取配置文件

  2、初始化配置参数，驱动函数

  3、让驱动 scan 当前所有的 bssid

  4、检查扫描的参数是否和用户设置的想否

  5、如果相符，通知驱动进行权限 认证操作

  6、连上 AP

- Wifi 驱动模块

厂商提供的 source,主要进行 load firmware 和 kernel 的 wireless 进行通信

- Wifi 电源管理模块

主要控制硬件的 GPIO 和上下电，让 CPU 和 Wifi 模组之间通过 sdio 接口通信

Wifi 工作步骤

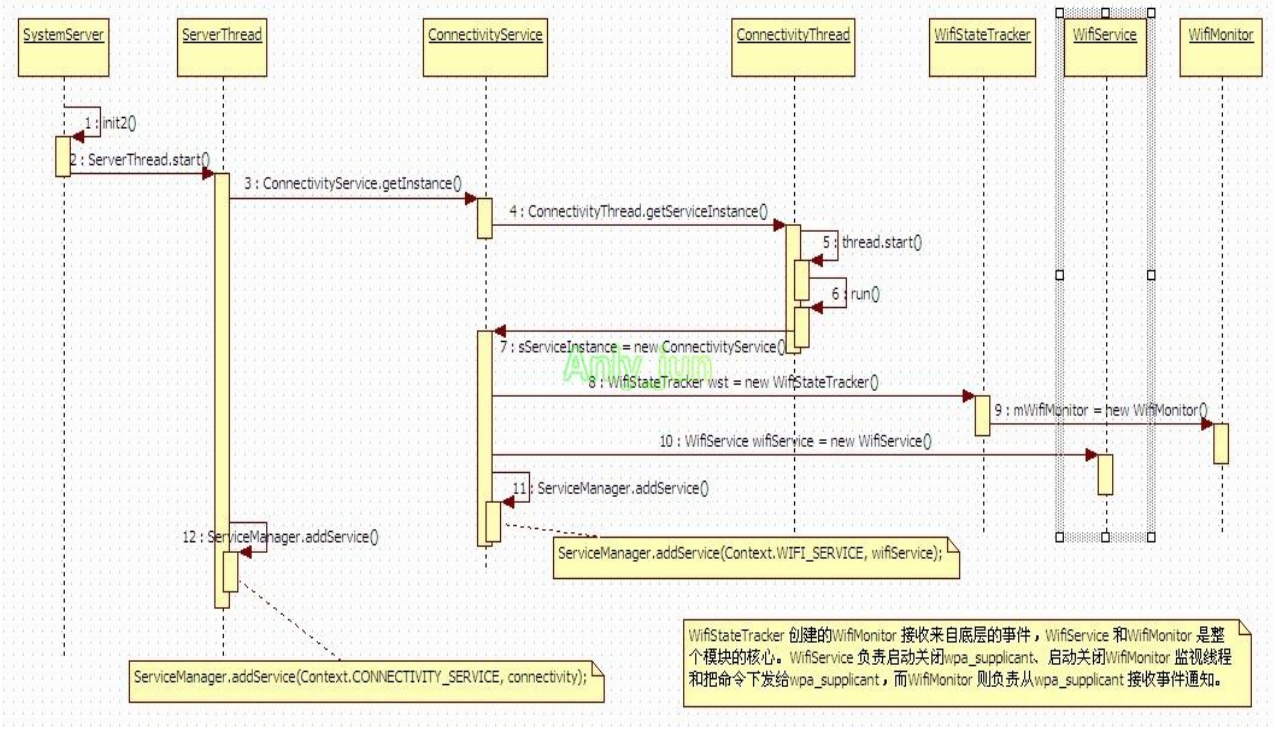
- Wifi 模块初期化
- Wifi 启动
- 查找热点（AP）
- 配置 AP
- 配置 AP 参数
- Wifi 连接
- IP 地址配置

Wifi 模块代码总结

- Wifi Application 代码
  packages/apps/Settings/src/com/android/settings/wifi
- Wifi Framework
  frameworks/base/wifi/java/android/net/wifi
  frameworks/base/services/java/com/android/server
- Wifi JNI
  frameworks/base/core/jni/android_net_wifi_Wifi.cpp
- Wifi Hardware
  hardware/libhardware_legacy/wifi/wifi.c
- Wifi tool
  external/wpa_supplicant
- Wifi kernel
  net/wireless drivers/wlan_sd8688 arch/arm/mach-pxa/wlan_pm.c

Wifi 模块的初始化:

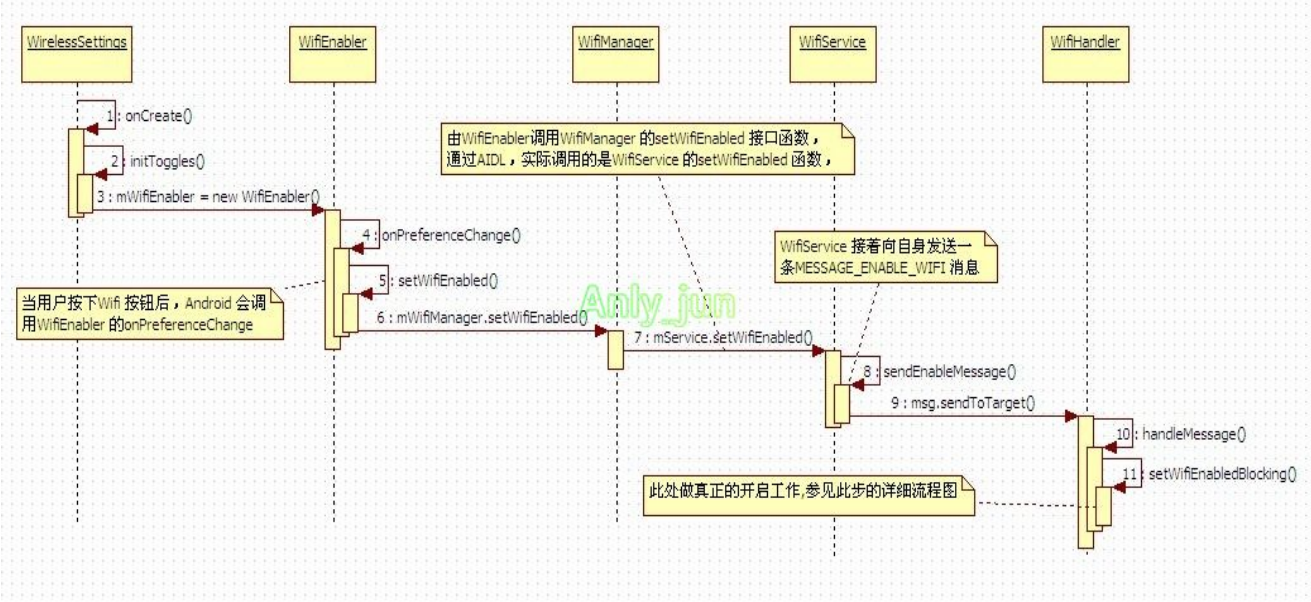
在 SystemServer 启动的时候，会生成一个 ConnectivityService 的实例，ConnectivityService 的构造函数会创建 WifiService，WifiStateTracker 会创建 WifiMonitor 接收来自底层的事件，WifiService 和 WifiMonitor 是整个模块的核心。WifiService 负责启动关闭 wpa_supplicant、启动关闭 WifiMonitor 监视线程和把命令下发给 wpa_supplicant，而 WifiMonitor 则负责从 wpa_supplicant 接收事件通知。

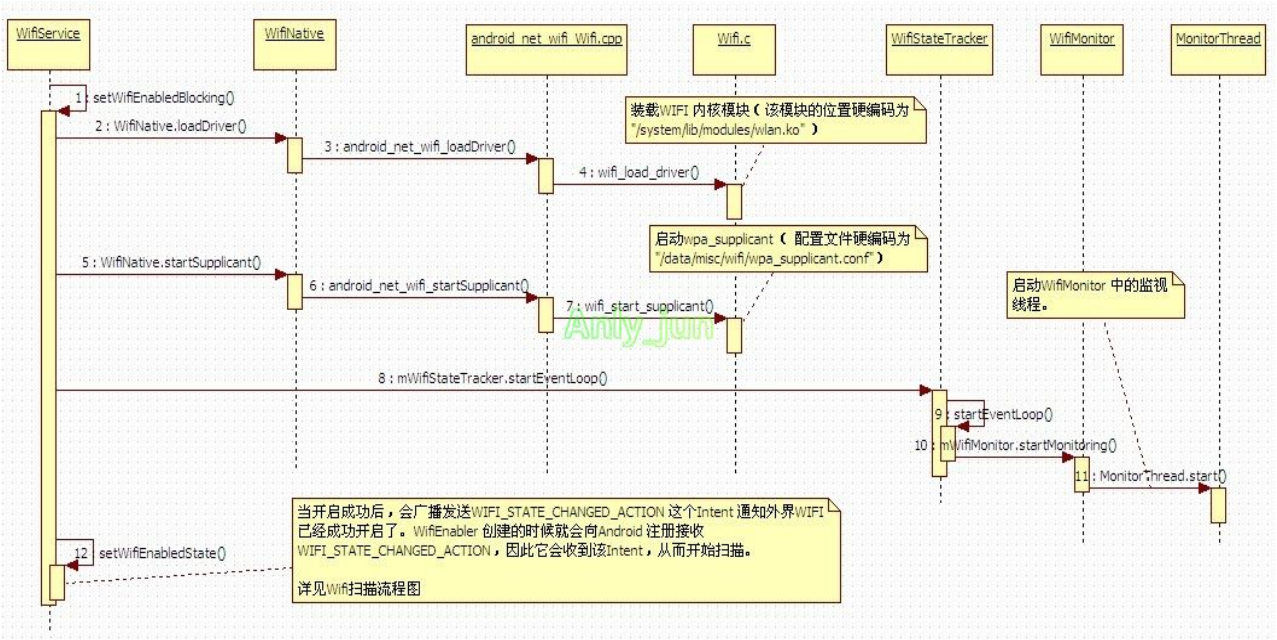


Wifi 模块的启动:

WirelessSettings 在初始化的时候配置了由 WifiEnabler 来处理 Wifi 按钮，

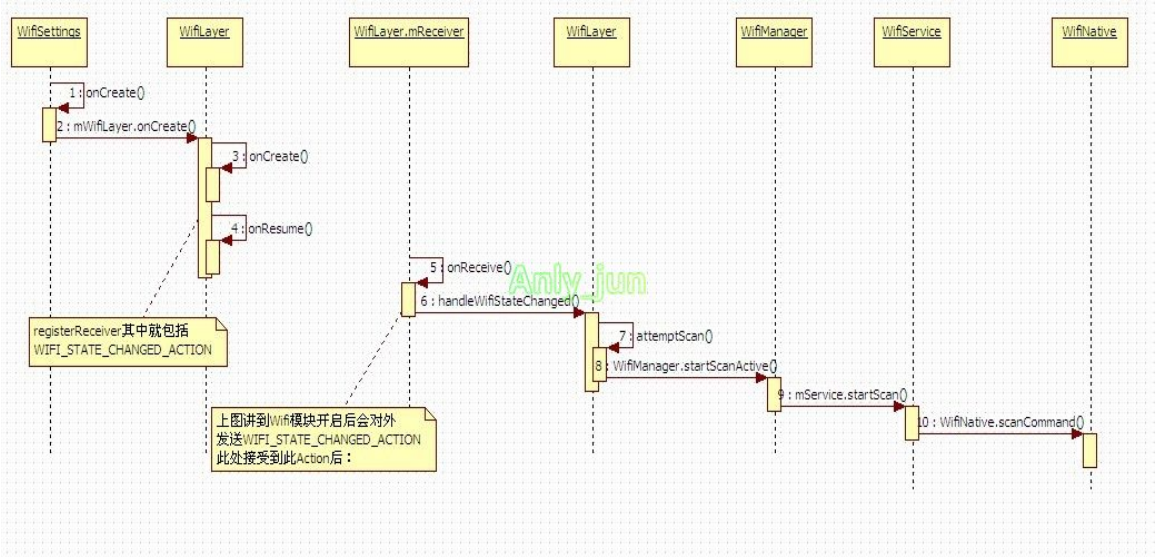
当用户按下 Wifi 按钮后，Android 会调用 WifiEnabler 的 onPreferenceChange，再由 WifiEnabler 调用 WifiManager 的 setWifiEnabled 接口函数，通过 AIDL，实际调用的是 WifiService 的 setWifiEnabled 函数，WifiService 接着向自身发送一条 MESSAGE_ENABLE_WIFI 消息，在处理该消息的代码中做真正的使能工作：首先装载 WIFI 内核模块（该模块的位置硬编码为"/system/lib/modules/wlan.ko"），然后启动 wpa_supplicant（配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



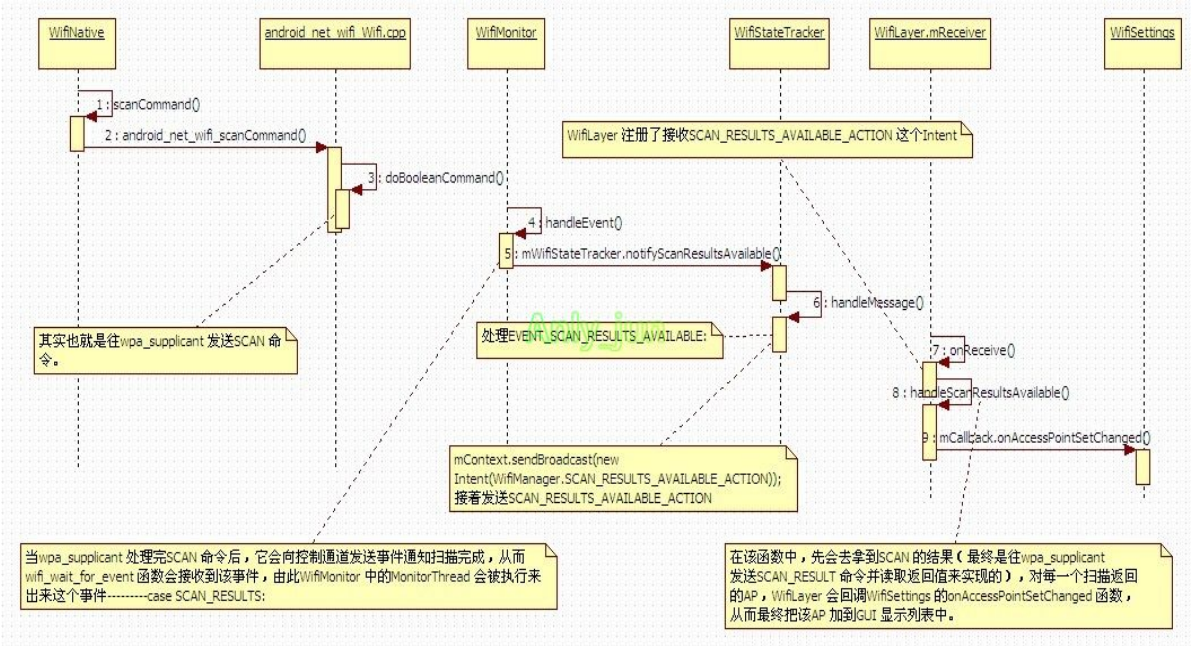


查找热点（AP）：

（Wifi 开启）中讲到 Wifi 模块开启后会对外发送 WIFI_STATE_CHANGED_ACTION。WifiLayer 中注册了 Action 的 Receiver。当 WifiLayer 收到此 Action 后开始 scan 的流程，具体如下

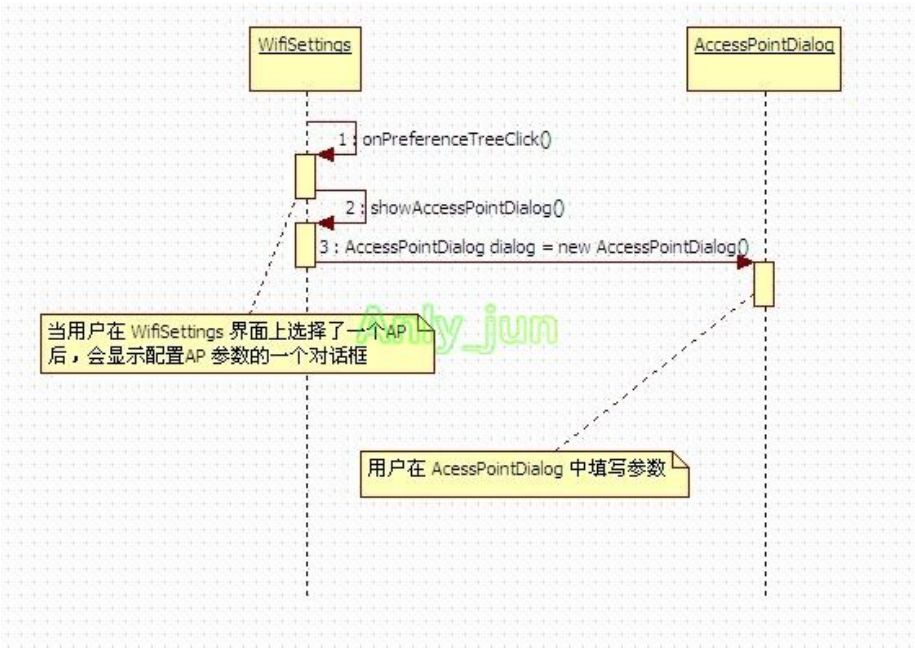


当 wpa_supplicant 处理完 SCAN 命令后，它会向控制通道发送事件通知扫描完成，从 wifi_wait_for_event 函数会接收到该事件，由此 WifiMonitor 中的 MonitorThread 会被执行来出来这个事件：

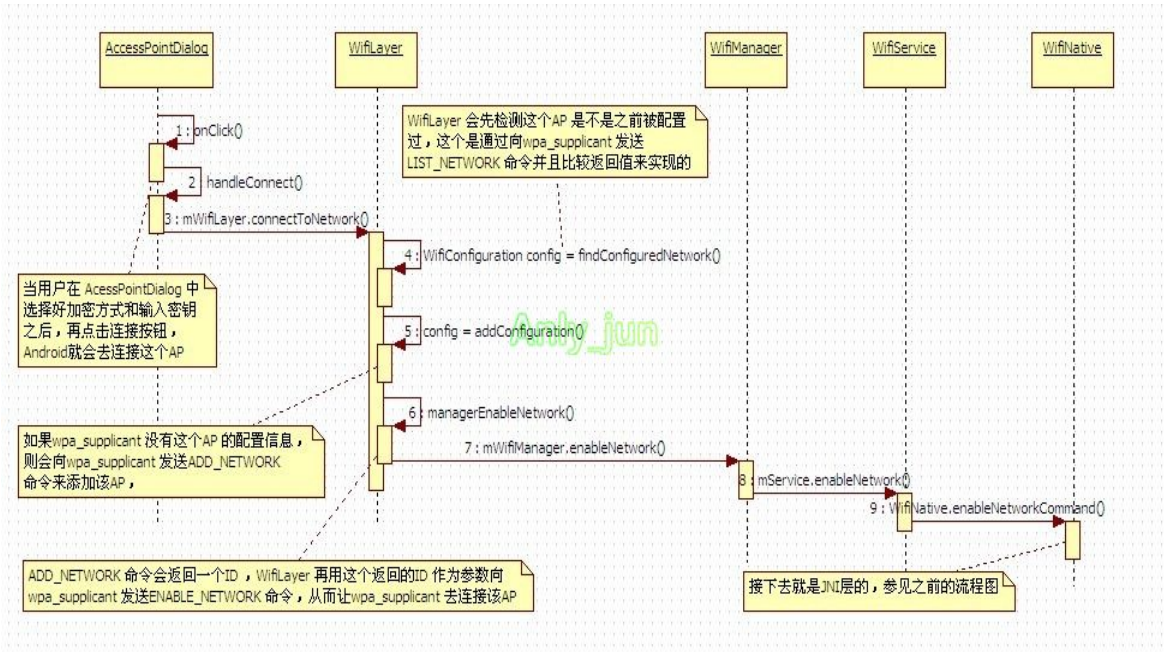


配置 AP 参数:

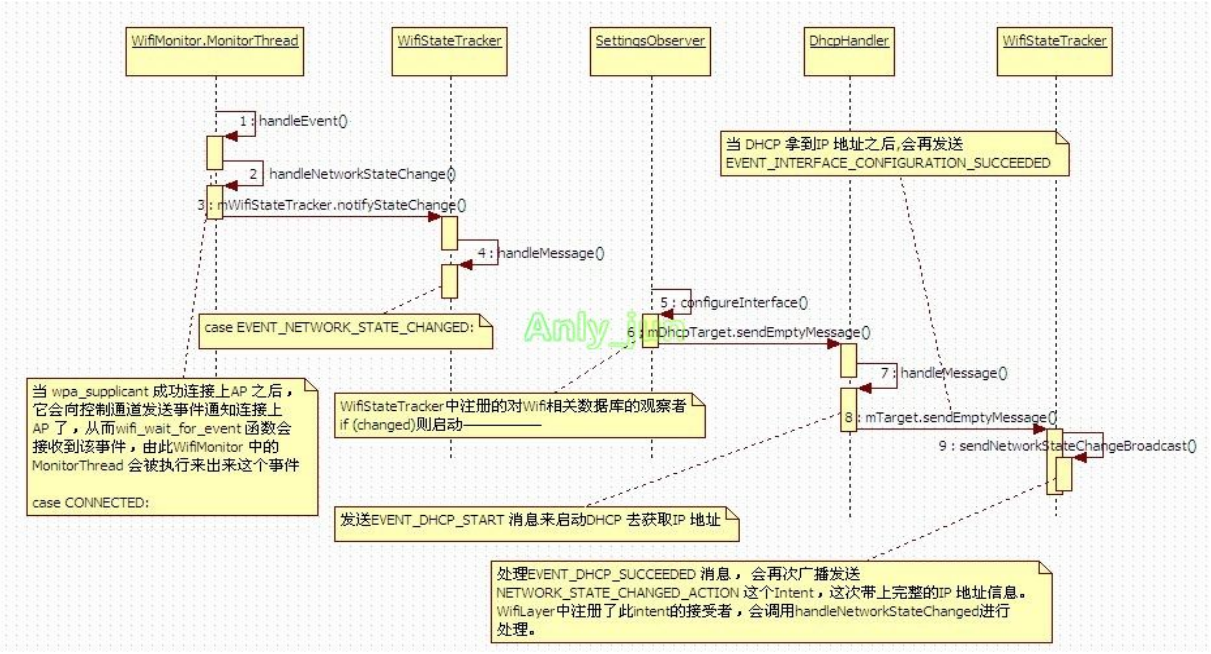
当用户在 WifiSettings 界面上选择了一个 AP 后，会显示配置 AP 参数的一个对话框：

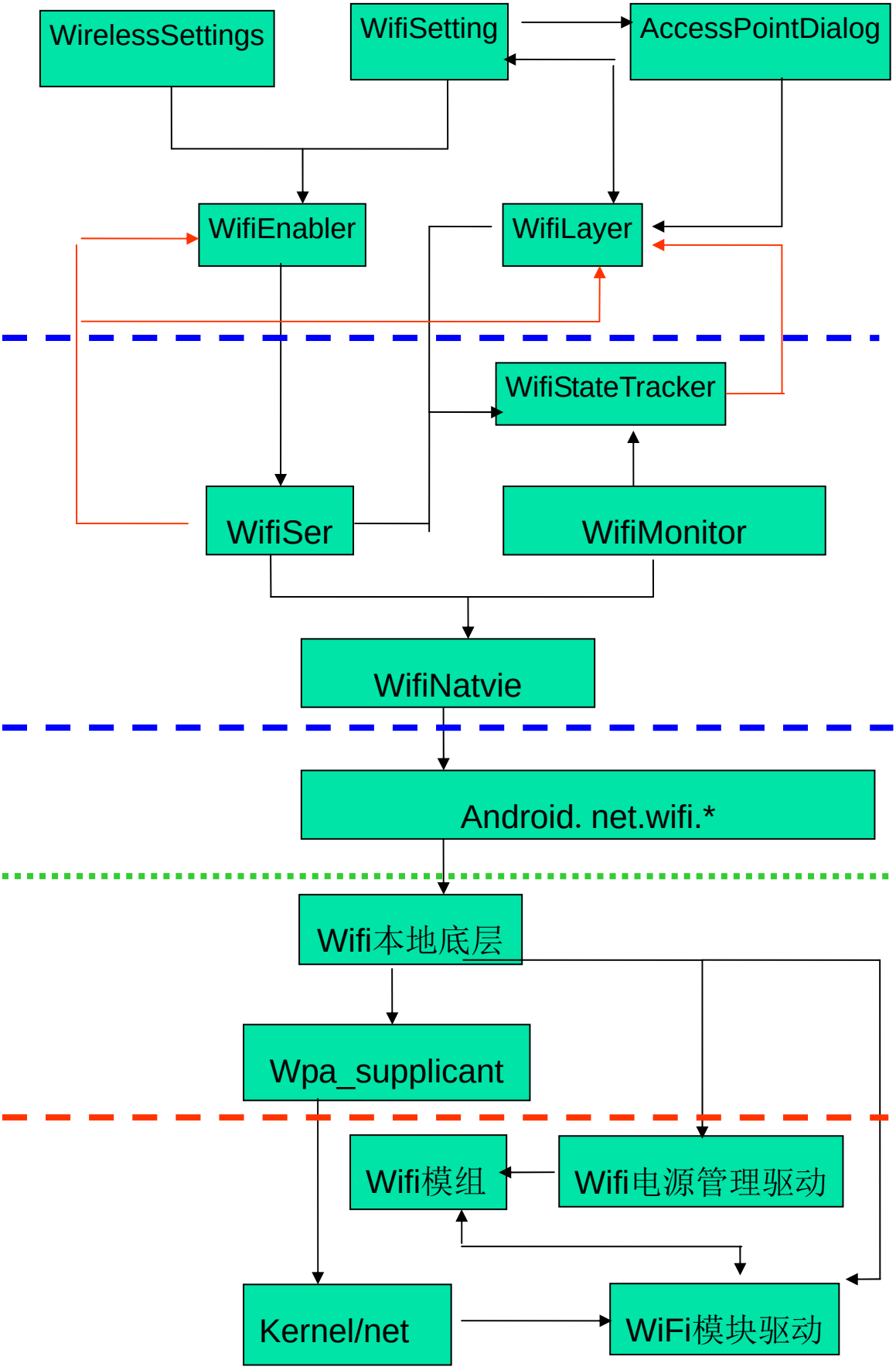


Wifi 连接:



IP 地址的配置:


WifiMonitor.MonitorThread
WifiStateTracker
SettingsObserver
DhcpHandler
WifiStateTracker
1 : handleEvent()
2 : handleNetworkStateChange()
3 : mWifiStateTracker.notifyStateChange()
4 : handleMessage()
5 : configureInterface()
6 : mDhcpTarget.sendEmptyMessage()
7 : handleMessage()
8 : mTarget.sendEmptyMessage()
9 : sendNetworkStateChangeBroadcast()
当 DHCP 拿到IP 地址之后,会再发送 EVENT_INTERFACE_CONFIGURATION_SUCCEEDED
case EVENT_NETWORK_STATE_CHANGED:
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 会被执行来出来这个事件
case CONNECTED:
WifiStateTracker中注册的对Wifi相关数据库的观察者 if (changed)则启动
发送EVENT_DHCP_START 消息来启动DHCP 去获取IP 地址
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。WifiLayer中注册了此intent的接受者，会调用handleNetworkStateChanged进行处理。


WirelessSettings
WifiSetting
AccessPointDialog
WifiEnabler
WifiLayer
WifiStateTracker
WifiSer
WifiMonitor
WifiNatvie
Android. net.wifi.*
Wifi本地底层
Wpa_supplicant
Wifi模组
Wifi电源管理驱动
Kernel/net
WiFi模块驱动