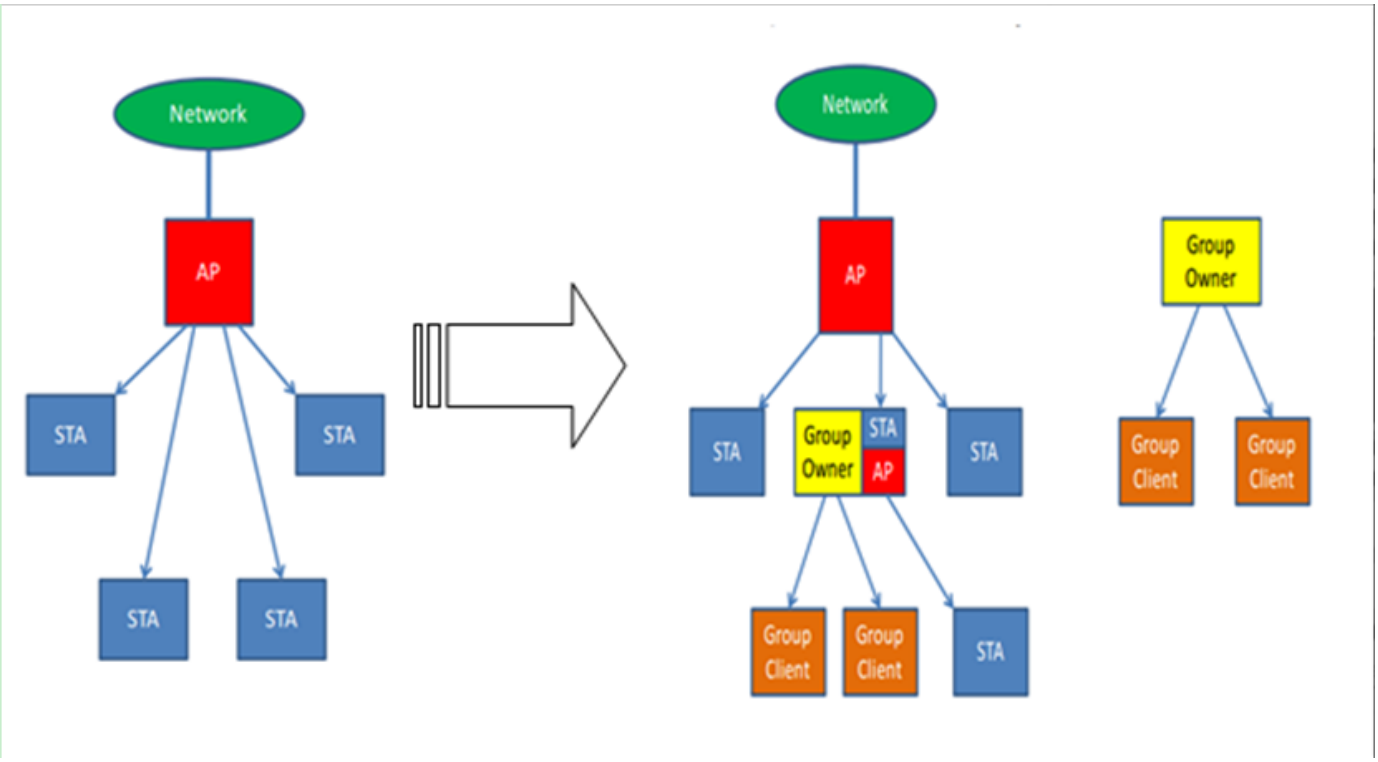
和菜鸟一起学android4.0.3源码之wifi direct的简单分析

2013-03-26 14:34 7643人阅读 评论(12) 收藏 举报

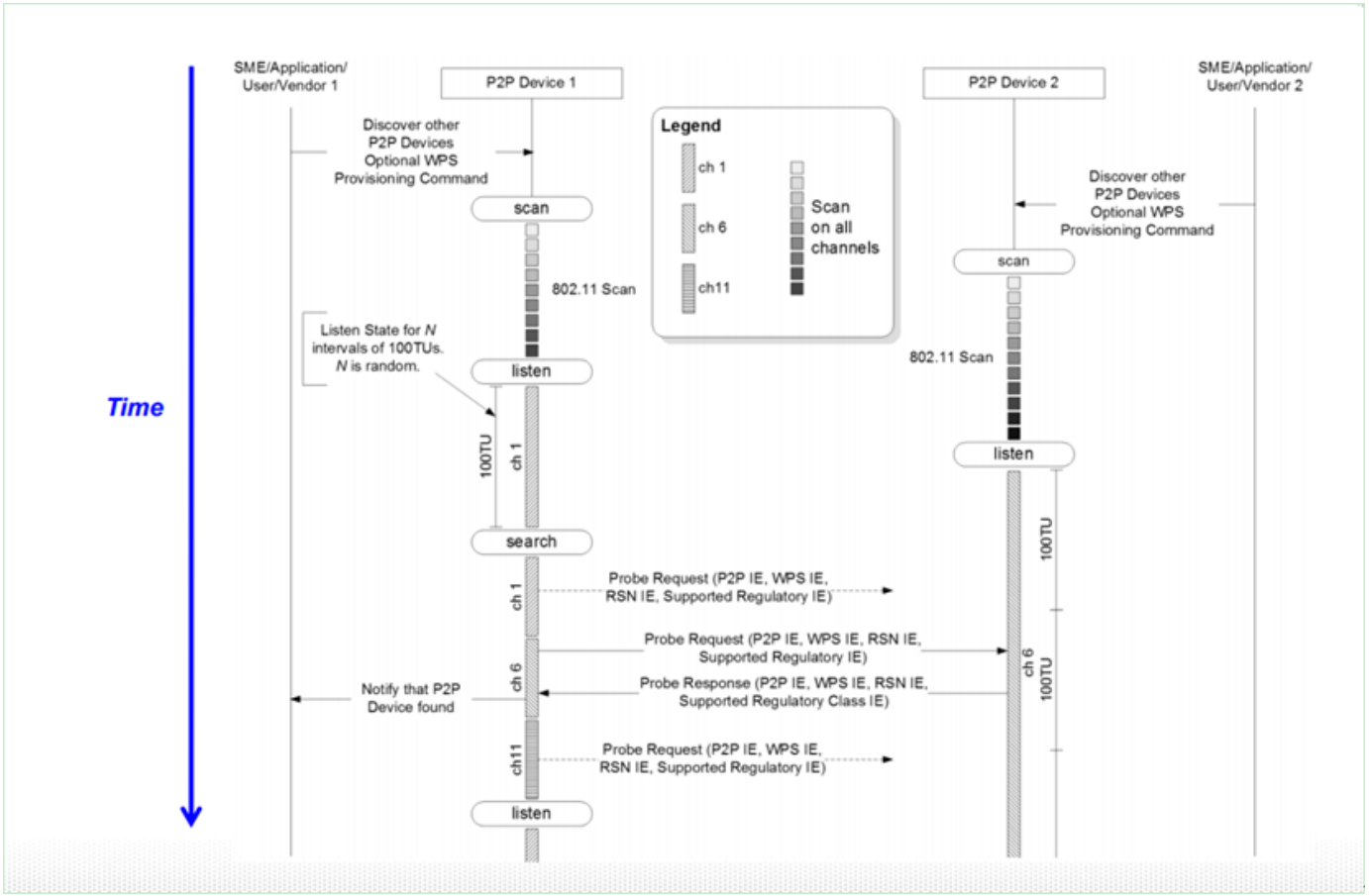
分类： Android源码学习之路（13）

关于**wifi direct**

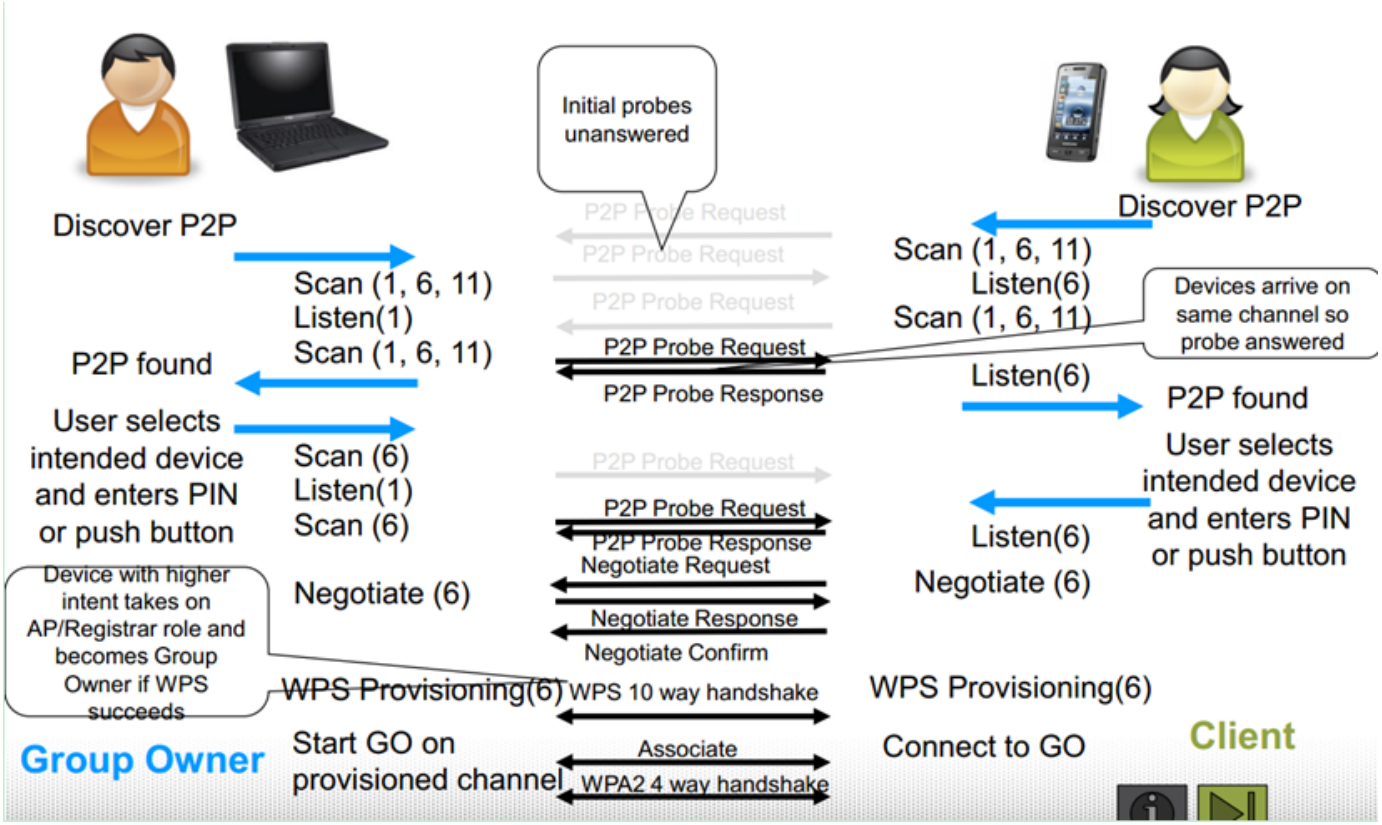
Wifi direct的连接



下面的图表示的是wifi direct的发现过程。

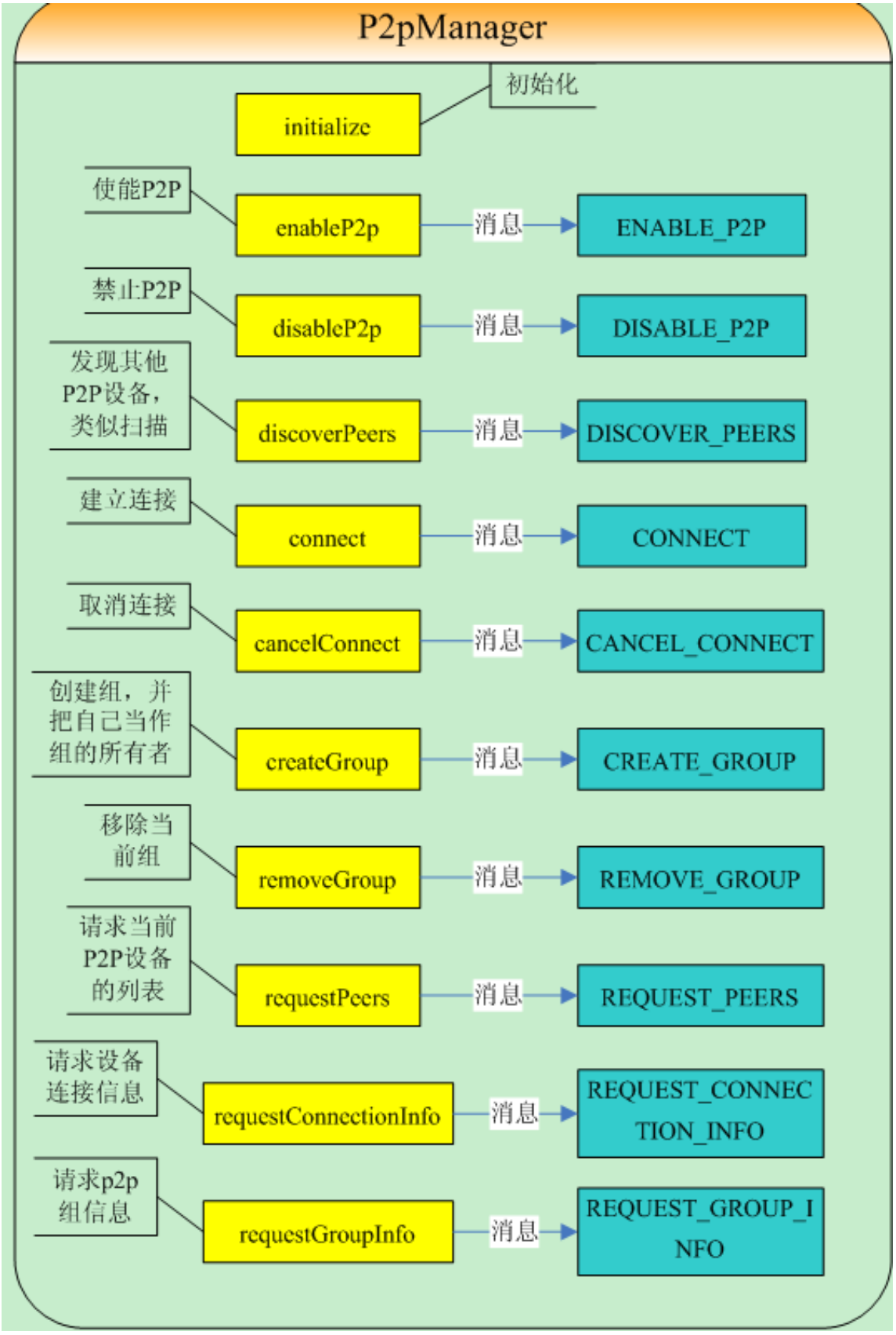


整个过程可以见下图。

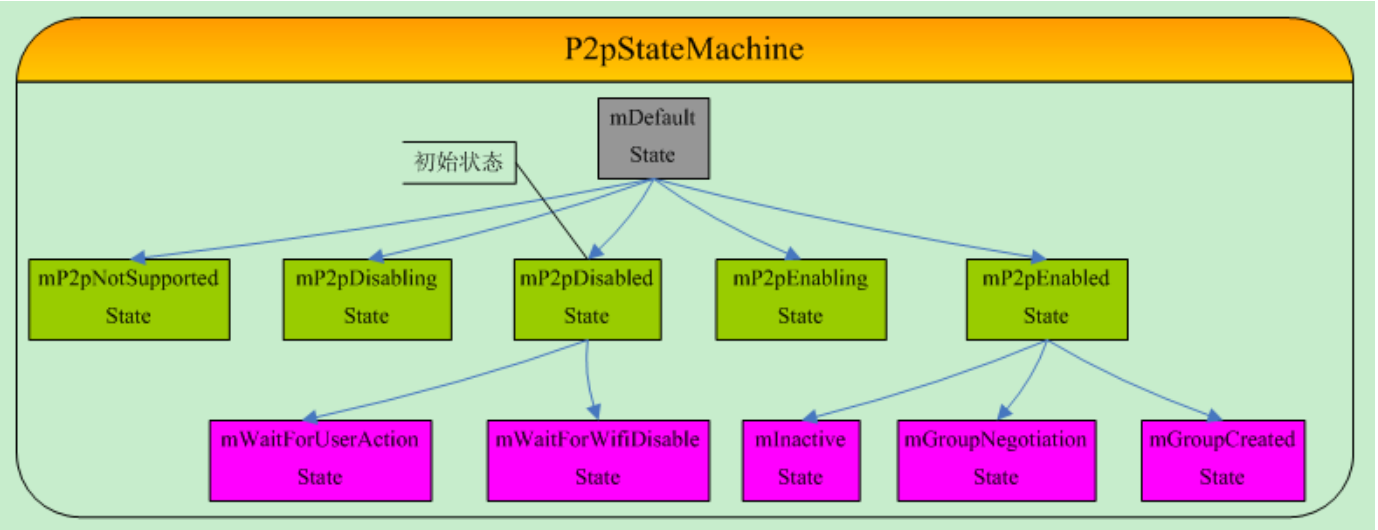


关于**Android**上的**wifi direct**

首先上层通过调用p2p manager的接口来实现p2p的使能，扫描，连接，群组的创建等等的功能。具体接口如下。



如上图所示，p2p的所有的处理都是在p2pservice中的p2pstatemachine中做处理的。具体的状态机的树形图如下所示：

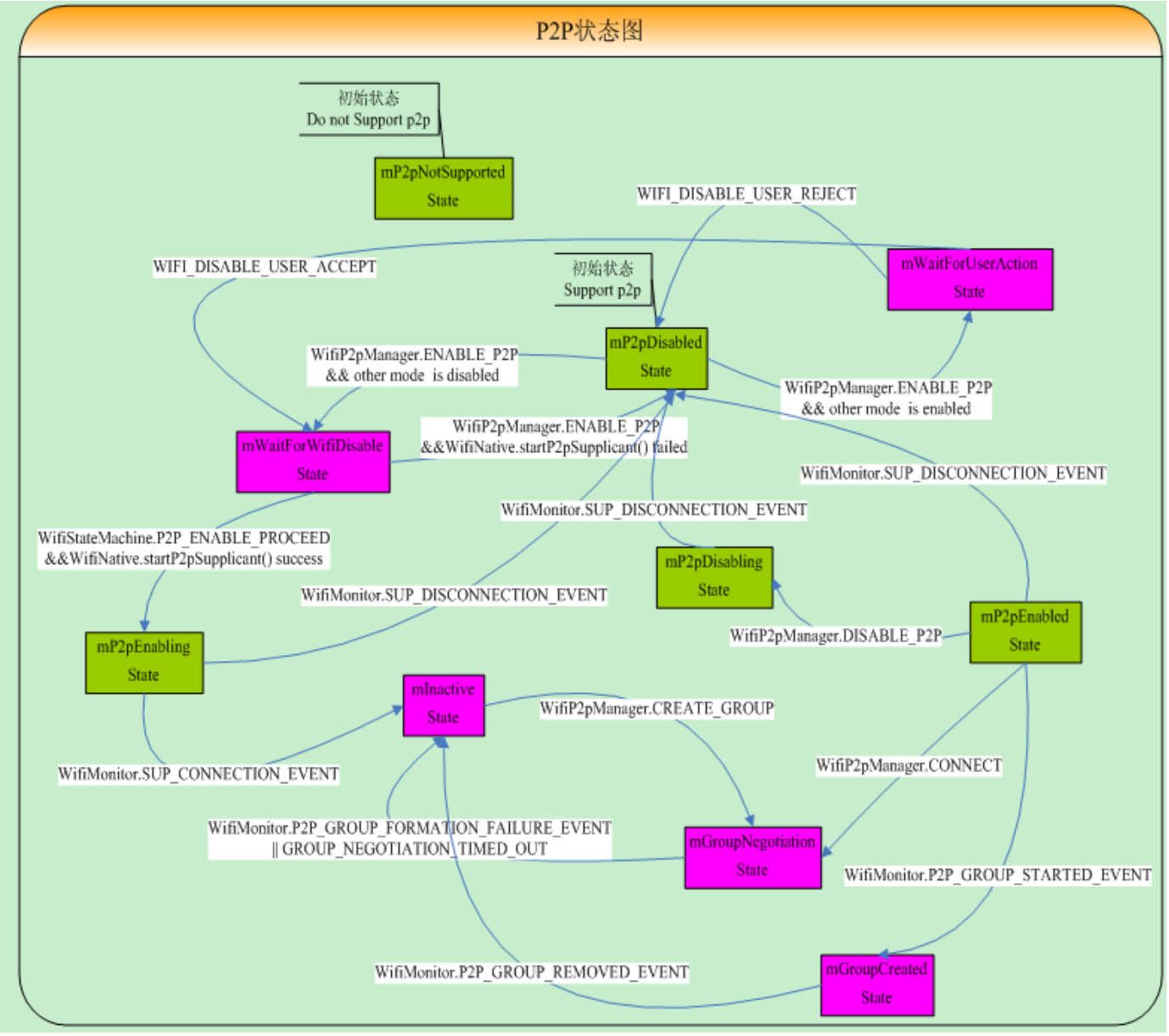


首先是P2pDisabledState，当使能后，也就是发送了WifiP2pManager.ENABLE_P2P，所以接着就调用了
mWifiChannel.sendMessage(P2P_ENABLE_PENDING);
transitionTo(mWaitForWifiDisableState);

其中P2P_ENABLE_PENDING会调用到wifistatemachine中的
caseWifiP2pService.P2P_ENABLE_PENDING:
　　mReplyChannel.replyToMessage(message,P2P_ENABLE_PROCEED);

会发回一个P2P_ENABLE_PROCEED的消息，接着状态转到WaitForWifiDisableState。所以他会处理case WifiStateMachine.P2P_ENABLE_PROCEED这条语句。执行
```
if(WifiNative.startP2pSupplicant()) {
    mWifiMonitor.startMonitoring();
    transitionTo(mP2pEnablingState);
}
```
这里就开启了p2p和wpa_supplicant的调用，然后wifi的monitor用来监听wpa_supplicant上报的事件。接着状态转移到了P2pEnablingState，在这里，如果p2p开启成功的话，也就是与wpa_supplicant的通信成功的话，wpa_supplicant就会上报一个事件，然后执行下面的语句：
```
caseWifiMonitor.SUP_CONNECTION_EVENT:
    logd("P2p start successful");
     transitionTo(mInactiveState);
     break;
```
这里p2p就使能完成了，状态进入到了InactiveState了。这时，如果要创建群组的话，那么就会调用到了caseWifiP2pManager.CREATE_GROUP:接着
```
if(WifiNative.p2pGroupAdd()) {
   replyToMessage(message,
```

WifiP2pManager.CREATE_GROUP_SUCCEEDED);
}
transitionTo(mGroupNegotiationState);
调用底层的p2p创建群组，然后状态转移到GroupNegotiationState 了。
在P2pEnabledState，状态中，如果调用了DISCOVER_PEERS，就会调用case WifiP2pManager.DISCOVER_PEERS:这条语句，接着调用到底层去处理。
if (WifiNative.p2pFind(timeout)) {
replyToMessage(message,
WifiP2pManager.DISCOVER_PEERS_SUCCEEDED);
}
如果成功了，那么就返回一个成功的消息。在P2pHandler中的handleMessage就会处理返回的消息了。
((ActionListener)listener).onSuccess();
具体的很多状态可以看下状态机的状态图。



A去连接B:

A device：
app层调用Java Manger的connect()函数(如果当前的设备不是p2p组的成员，那么这个函数就发送groupnegotiation请求；如果这个设备已经是p2p组的成员，或者自己创建了一个组通过createGroup(),那么这个函数就发送邀请请求)，这个函数有个参数是WifiP2pConfig config，它包含了例如自己的MAC，用什么方式连接等一些信息。 通过JAVA Manger的中转，会到达JAVA Service层的P2pEnabledState状态下，这个状态下的处理函数收到WifiP2pManager.CONNECT后，通过调用WifiNative.p2pConnect(mSavedConnectConfig, join)，一层层往底层调用，调用会根据情况是否返回一个PIN。这样一个group negotiation请求就发送出去了。 如果返回PIN则通过
notifyWpsPin(pin, mSavedConnectConfig.deviceAddress)，会以AlertDialog的方式显示出来。
B device：
JAVA Service层处在InactiveState状态下，收到了来自底层的WifiMonitor.P2P_GO_NEGOTIATION_REQUEST_EVENT消息，该消息表示有设备请求GO协商。 收到消息
notifyP2pGoNegotationRequest(mSavedGoNegotiationConfig)函数被调用，该显示AlertDialog，用于用户同意或者取消GO 协商，如果对方是PIN方式，那么AlertDialog上面会有输入框显示，如果是PBC那么输入框就隐。

关于jni下面的调用，与wifi的类似，这里就不多做讲解了。
可以参见《和菜鸟一起学android4.0.3源码之wifi的简单分析》。