# Android平台简介

## Android架构介绍

[android 10 wifi architecture](https://blog.csdn.net/qq_39040377/article/details/104497354?spm=1001.2014.3001.5501)

## Android编译指令

Android镜像编译

->source build/envsetup.sh

->lunch

->make -j16

Android内核编译

->make bootimage

Android 单个模块编译

->mm/mma

## Android调试指令

工具adb 

adb root----------------->获取root权限

adb remount----------->重新挂载

adb shell--------------->shell命令

1.app log

adb logcat –v time > c:\appLog.txt

2.radio log

adb logcat –v time –b radio >c:\radioLog.txt

3.kernel log

adb shell "cat /proc/kmsg" > kernel.log

4.导出当前缓存的kernel log

adb shell dmesg

5.实时查看kernel log

adb shell kmsgcat

tcpdump抓取tcp报文

tcpdump -i any -s 0 -w /data/vendor/wifi/file.pcap

adb pull /data/vendor/wifi/file.pcap ./

//注：

//-i any 表示抓取所有接口（waln0、p2p0等）

//data/vendor/wifi/file.pcap 为设备端存储的路径

手动启动网卡

1.cd vendor/bin/hw/

./wpa\_supplicant -iwlan0 -Dnl80211 -c/data/misc/wifi/wpa\_supplicant.conf -O/data/misc/wifi/sockets -ddd

2.重新打开一个adb shell窗口

wpa\_cli -i wlan0 -p/data/misc/wifi/sockets -s/data/misc/wifi/sockets

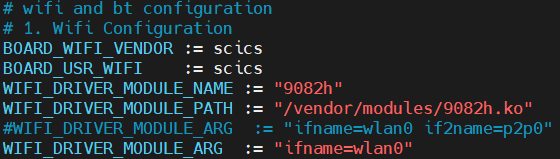
## Android驱动适配

->[卡维恩适配](https://app.yinxiang.com/fx/dd4de26d-b4ab-4e3d-a3f5-ccd5de2d494e)

->[橙视适配](http://182.150.24.204:18080/redmine/issues/692)

->全志平台

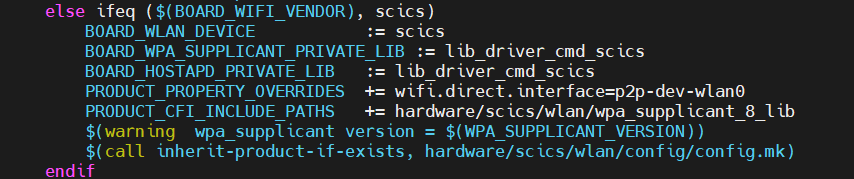
softwinner/eros-p1/BoardConfig.mk



1.配置驱动厂商

2.配置加载参数：

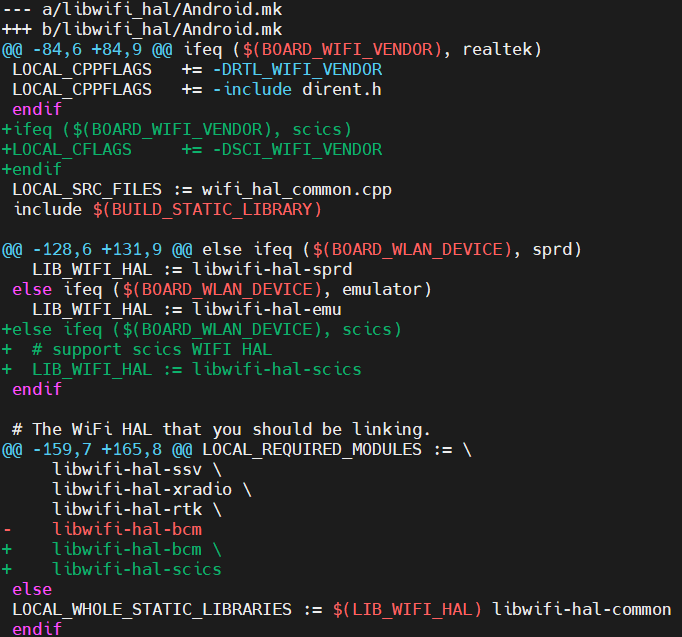
softwinner/common/config/wireless/



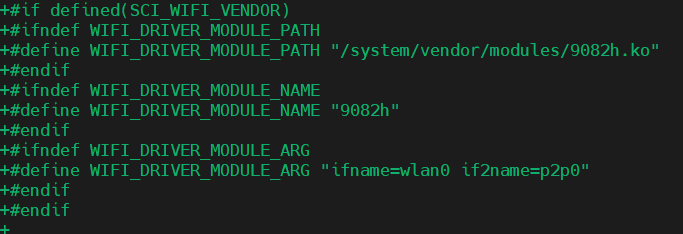
1.配置hal层使用的库

2.配置wifi直连的网口

frameworks/opt/net/wifi/libwifi\_hal/Android.mk



frameworks/opt/net/wifi/libwifi\_hal/ wifi\_hal\_common.cpp



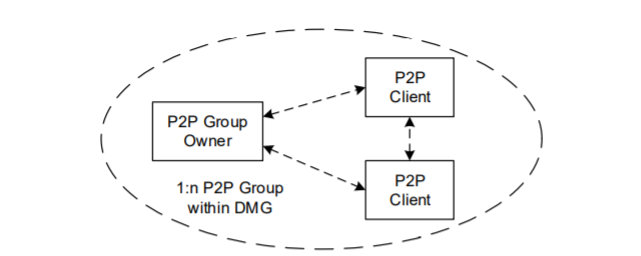
# P2P介绍

## P2P架构

P2P Device

P2P Group Owner : GO

P2P Client : GC



[p20]

## P2P discovery

->功能：p2p设备相互发现并形成链接

->包含如下技术：

-Device Discovery

不同的p2p设备在共同的信道交换信息相互发现

-Service Discovery

-Group Formation

决定哪个设备成为go，哪个设备成为gc并形成一个新的p2p group

-P2P Invitation

激活一个永久的p2p group，或者邀请一个p2p device 加入已存在组

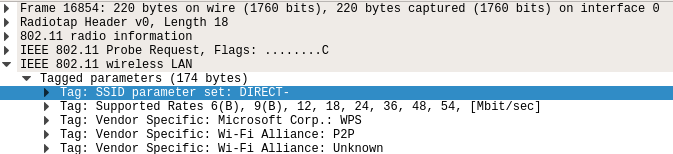
#### Device Discovery

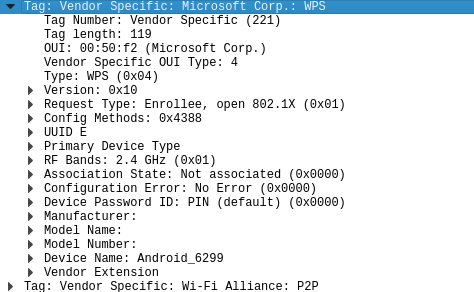
->搜索阶段：scan -> find : probe request

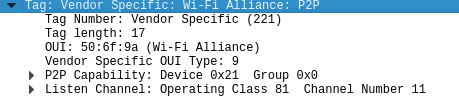
->被发现阶段： listen sate -> 1/6/11: probe response

此过程是通过probe request 和probe response 交换设备各自信息

##### probe request 帧







1.SSID : 其值必须为DIRECT-，它是p2p协议中规定的称为 P2P Wildcard SSID

2.wps ie:的作用： [wps介绍](https://blog.csdn.net/zhuhuan_5/article/details/107593368)

Device Name: 发送设备的名字

Primary Device Typ：指定接受者的类型

3.P2P ie: OUI=0x50-6f-9a-09

P2P capability：当前的p2p设备所支持的特性

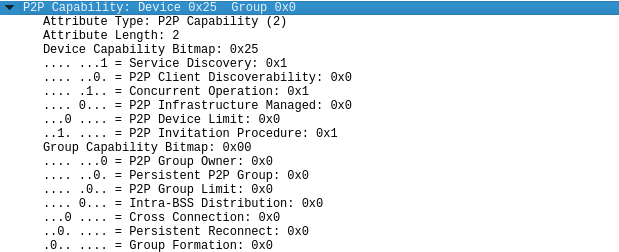
->p2p client discoverability:

* P2P Device A以client 加入了一个P2P Group
* P2P Device B定向搜索P2P Device A。由于Device A已经扮演了Client的角色，所以它不会回复Probe Response。不过，Group 1的GO却存储有当前与它关联的Client信息，即Group 1的GO了解Device A的信息。
* 如果Device A支持Client Discoverability，那么Group 1的GO、Device A以及Device B将借助Device Discoverability Request/Response帧来获取相关信息

->concurrent operation: 是否支持P2P和STA同时工作

->p2p device limit: 此p2p设备只能加入一个p2p group

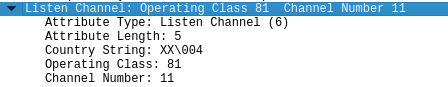
->p2p invitation procedure: 是否支持邀请流程



->intra-bss distribution: group 是否为client提供数据分发服务。

->group formation: provision 阶段是否扮演GO角色

Listen Channel：当前的p2p 设备在listen sate阶段使用的频段



->Country string: 国家码\ operating Class

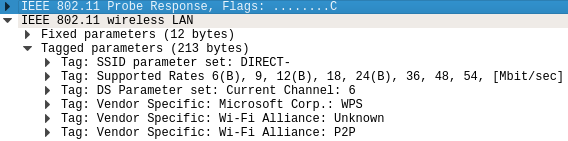
->operating Class：Listen state阶段使用的频率波段类别

->Channel Number: Listen channel 的值

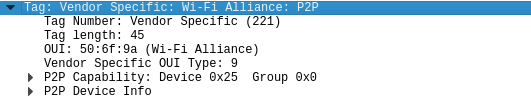
Probe request最基本的要求：

1. SSID 必须是 DIRECT-
2. P2P IE必须包含
3. Mac帧头中：Destination Address必须是广播地址或者p2p devices的地址，BSSID域必须为广播地址

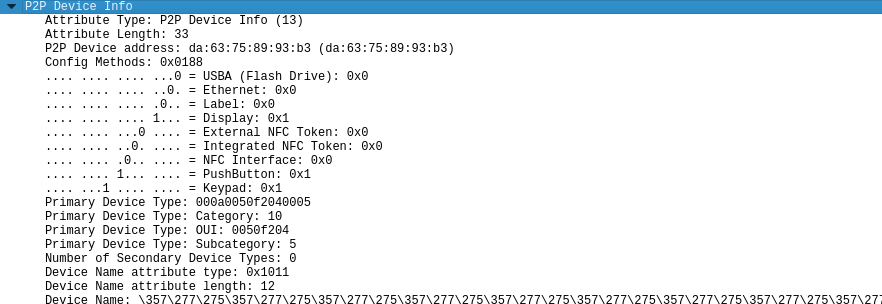
##### probe response 帧



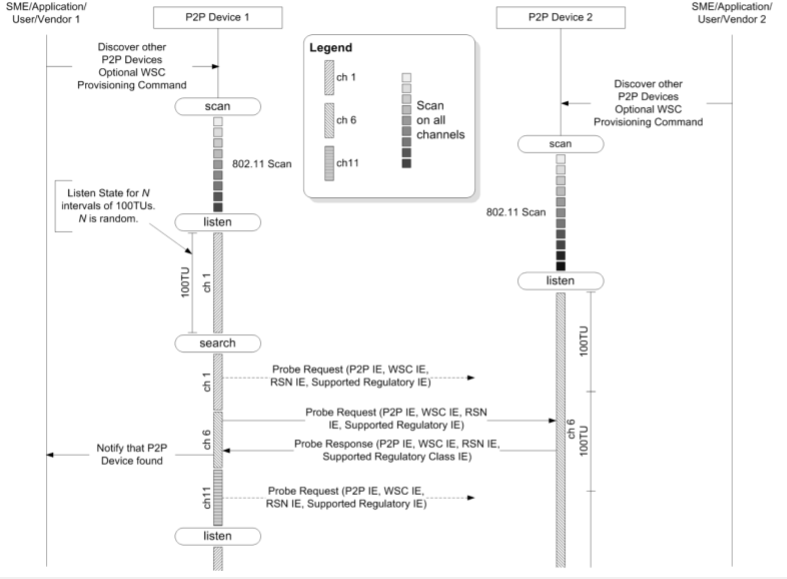




Response 帧的内容和request帧的内容很相似，p2p IE中多了一个P2P Device Info



##### Device Discovery 流程图



[p31]

->Scan

在各个频段上发送request帧,此阶段不会处理其他设备的request帧仅仅自己发送

->listen

停留在Listen channel，当接收到其他设备的request帧有p2p IE，会在此信道回复rsp

Listen channel： 从Social Channels(1/6/11)随机选择一个，并且确定后不能再更换

->search:

在social channel中发送带有p2p IE的reqst帧

#### Group Formation

通过组协商流程可以建立一个p2p group,此流程分为如下两个阶段

[Provision Discovery]

Go Negotation：GO与GC的角色协商

Provisioning：借助WSC交换安全配置信息

此过程中的交互都是使用的 P2P Public Action 帧

利用帧携带的请求信息，接受对方做处对应的处理

Action 帧的结构：Category + Action Detail

Category表示不同类型的Action帧

0: Spectrum Management

4: Public , 此类型就是p2p 协议使用的

5: Radio Management

127：Vendor Specific，和具体厂商有关，p2p协议也会使用



Category 0x04 P2P Public Action

Action filed 0x09 Public Action的一个子类型

Subtype 0x07 不同的Action帧

Dialog Token req/rsp交互标识



###### Provision Discovery

作用：交互双方使用WSC方法

流程：->Provison Discover Request 设置一个Config Method方法

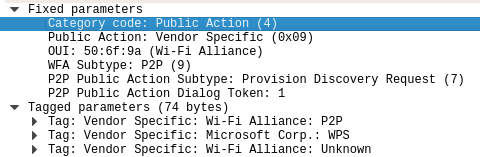
->接受到req帧的设备将根据设置的方法触发对应的操作(如显示pin码等)

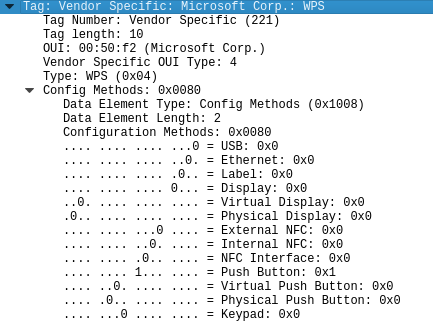
->Provision Discovery Response 将会对req配置的方法进行回应，正常会回复req配置的方法，如果不支持req的方法将会回复为null

总结：在PD流程中主要起作用的是WPS IE

在我的理解中，整个组协商流程其实是嵌入在WPS的流程之中的，PD流程确定wps流程方法后，马上展开组协商确定好角色，继续进行wps流程链接成功

->Provison Discover Request





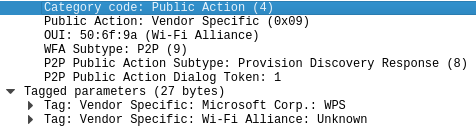
->PD request帧WPS IE中的Config Method属性设置想要使用的WSC配置方法，图中设置的是Push Buton

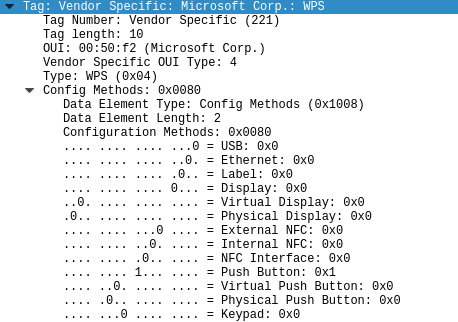
->如果接受者也支持Push Button，则回复的rsp帧中的配置方法也将是Push Button

->如果接受者不支持Push button ,则回复rsp帧Config Method的值将置0

当rsp帧的Config Method的值为0，req发送者将会配置一种新的方法从新发起请求

->Provison Discover Response





WSC连接流程时需要用户参与来配置一些信息的(输入pin码)

PD流程本身不属于组协商，它发生在Group Formation之前，提前邀请用户输入wsc安全配置所需要的信息，这些信息用于provsioning流程，避免了provsioning流程时间过长超过15秒

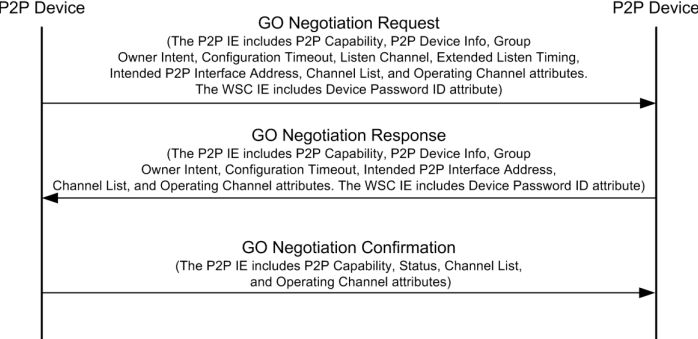
###### Group Owner Negotiation

->标准的组协商流程包含3帧

->GO Negotiation Request

->GO Negotiation Response

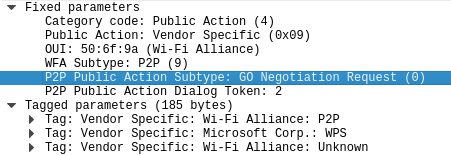
->GO Negotiation Confirmatiom

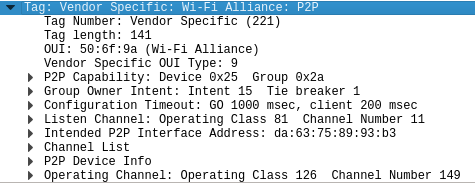


[p47]

此流程交互双发进行信息交换，决策出P2P网络中的GO和GC角色，主要是P2P IE

->GO NEGO Request帧

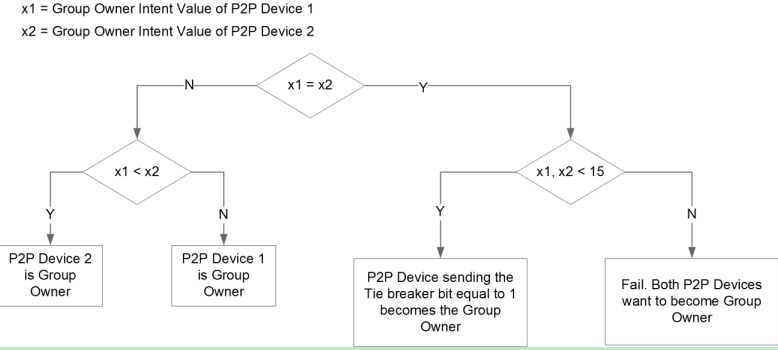




->GO Intent: 表示成为GO 角色意愿的强烈度，取值范围0~15，值越高越有机会成为GO

->Tie breaker: 随机取值0/1，当协商双方intent相同，此位为1则成为GO

如果协商双方intent的值都为15，将协商失败



->Configuration Timeout: 表示切换成相对应的角色的超时时间

->Channel List: 表示发送方设置的国家码和支持的WiFi频段信息

-> Intended P2P Interface Address:表示p2p设备加入group后使用的mac地址

-> operating channel: 表示期望p2p group工作的频率

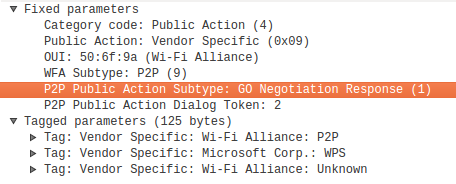
[注]：

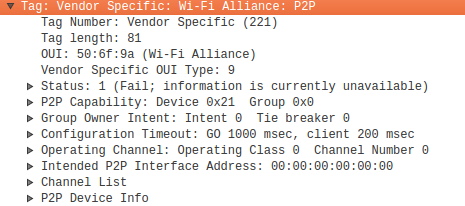
P2P设备接收到GON Request帧后，设备会弹框让用户选择接受或者拒绝

接受的话：协商流程继续进行

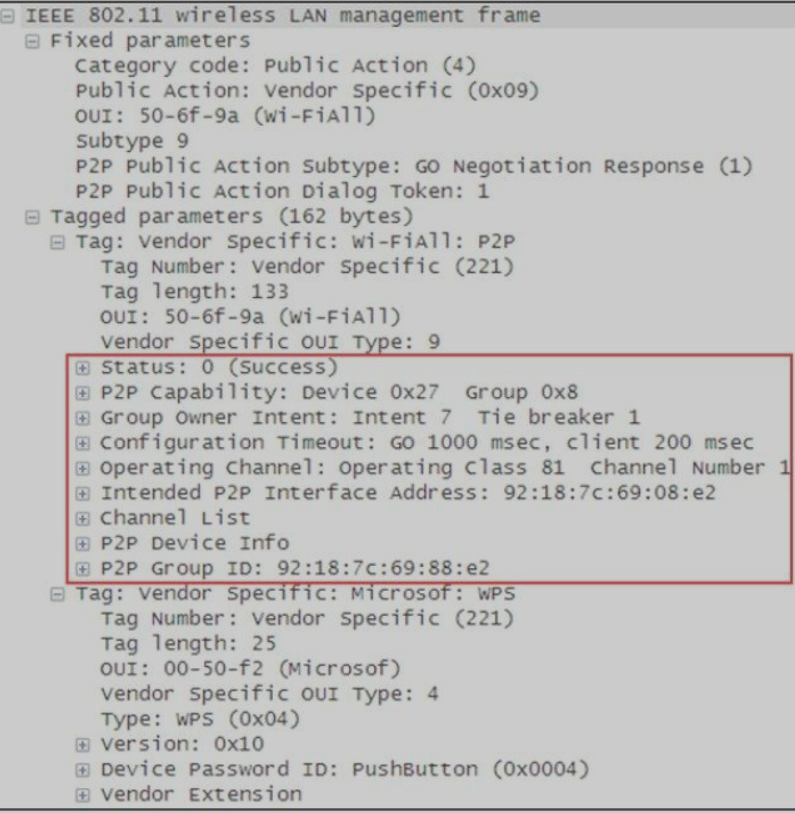
拒绝：协商流程终止

->GO NEGO Response帧





->Status: 表示对接搜到的req帧的处理结果， 0表示成功，其余表示失败



Status: 0表示处理成功

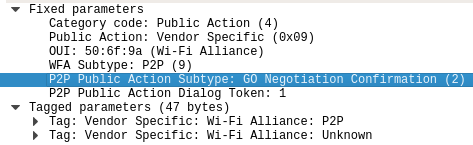
P2P Group ID: 用于标识唯一表示一个P2P Group包含两个属性 p2p devicxe address + SSID

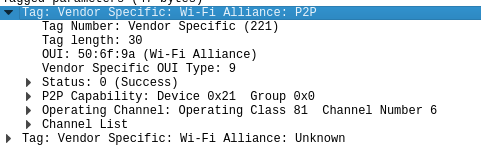
SSID: 起前缀必须为”DIRECT-“

仅P2P group owner的rsp会有Group ID

->GONE Confirmation

此帧是对rsp帧的确认，GO角色的设备回复此帧必须包含Group ID属性

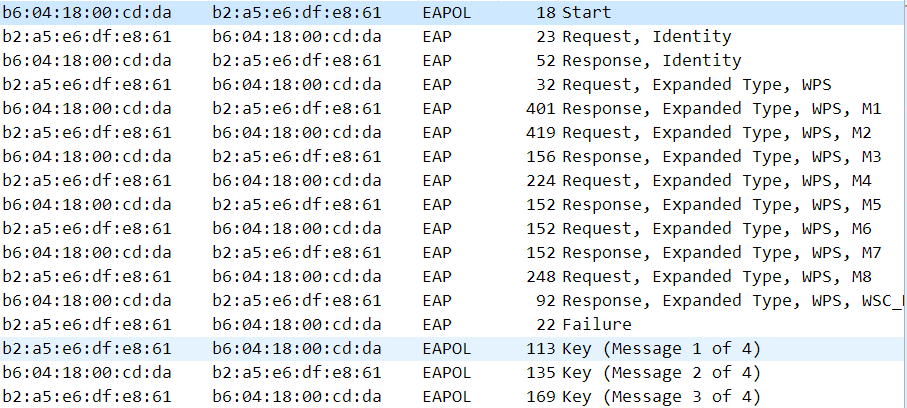




#### Provisioning

此过程就是WSC流程，利用数据包交换安全配置信息，组建成完整的P2P网络

[provisioning](../../log_file/record/wfd-20210927%20-oppo-ok.pcap)



对于整个组协商过程我们驱动的打印：

1. wf\_wlan\_cfg80211\_remain\_on\_channel(p2p0) ch:1 duration:307, cookie:0x18
2. wf\_wlan\_cfg80211\_remain\_on\_channel, set ro ch timer, duration=307
4. -> cfg80211\_remain\_on\_channel
5. //停留在"listen\_channel"进入监听流程，接收P2P\_GO\_NEGO\_REQ帧
7. WL\_Rx:cur\_ch=1
8. WL\_Rx:P2P\_GO\_NEGO\_REQ, dialogToken=2, intent:15+, listen\_ch:6, op\_ch:1, ch\_list:1-13, full\_ch\_in\_p2p\_handshake:0
10. ->cfg80211\_rx\_mgmt
11. //listen\_channel 接受到P2P\_GO\_NEGO\_REQ帧，通过cfg80211\_rx\_mgmt，上报给框架
13. WL\_Tx:tx\_ch=1, da=56:dc:1d:34:fa:cc
14. WL\_Tx:P2P\_GO\_NEGO\_RESP, dialogToken=2, intent:0-, status:1, op\_ch:1, ch\_list:
16. ->\_cfg80211\_Mgmt\_Tx
17. //在listen—channel调用.mgmt\_tx = \_cfg80211\_Mgmt\_Tx 回复resp帧
18. =============================================================================
19. wf\_wlan\_cfg80211\_remain\_on\_channel(p2p0) ch:6 duration:500, cookie:0x1e
20. wf\_wlan\_cfg80211\_remain\_on\_channel, set ro ch timer, duration=500
21. WL\_Tx:tx\_ch=6, da=56:dc:1d:34:fa:cc
22. WL\_Tx:P2P\_GO\_NEGO\_REQ, dialogToken=1, intent:7-, listen\_ch:1, op\_ch:1, ch\_list:1-13, full\_ch\_in\_p2p\_handshake:0
24. ->cfg80211\_remain\_on\_channel
25. //将信道切换到peer端的listen\_channel
26. ->\_cfg80211\_Mgmt\_Tx
27. //在peer端的listen\_channel信道发送P2P\_GO\_NEGO\_REQ
29. WL\_Rx:cur\_ch=6
30. WL\_Rx:P2P\_GO\_NEGO\_RESP, dialogToken=1, intent:15+, status:0, op\_ch:1, ch\_list:1-13
32. ->cfg80211\_rx\_mgmt
33. //peer端的listen信道接收到P2P\_GO\_NEGO\_RESP
35. WL\_Tx:tx\_ch=6, da=56:dc:1d:34:fa:cc
36. WL\_Tx:P2P\_GO\_NEGO\_CONF, dialogToken=1, status:0, op\_ch:1, ch\_list:1-13
38. ->\_cfg80211\_Mgmt\_Tx
39. //发送P2P\_GO\_NEGO\_CONF帧，协商完成

#### P2P Invitation Procedure

邀请流程发生在如下情况：

->GO 邀请P2P Device 以GC角色加入自己的P2P Group

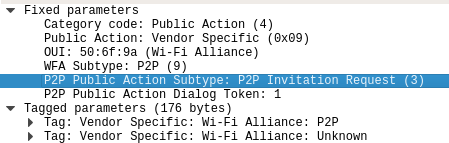
->GC 邀请P2P Device 加入当前的组中以便使用一些服务

->用于激活一个永久的P2P Group

整个流程可分为：

1. 使用WPS获得身份凭证，PD和WPS都在p2p 组GO的operating channel
2. 如果P2P Device已经provisioned，就链接到P2P组

###### P2P Invitation Request帧





不同邀请情况属性的配置有区别

1.被一个P2P Group里面的设备邀请

->Invitation Flags: 0

->Timeout:0

->Operating channel: 指明operating channel，必须使用这个

2.被Persistent P2P Group中的另一个设备邀请用于激活

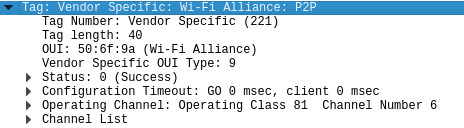
->Invitation Flags: 1

->Timeout:表示当接受到rsp后GO开启的时间限制

->Operating channel: 预示operating channel将要使用信道，最终不一定要使用这个

###### P2P Invitation Response帧





Status: 0表示成功接受邀请，其他的值为错误码

在两个设备没有共同的channel时，status 会设置为8

Fail：information is currently unavailable

120s以内上层有回应，将重新发起req

上层拒绝连接，被邀请的经不在有任何动作

Chanel List：必须是request的子集

Rsp成功：

->有时p2p client在预测的operating channel 无法探测到GO的存在，GO可能是因为别的原因在使用了不同的信道

## P2P of wpa\_supplicant 初始化

->nl80211\_mgmt\_subscribte\_non\_ap

//注册了P2P两种类型帧的监听事件

P2P Public Action帧监听事件：根据P2P规范，目前使用的均是802.11 Public Action帧，即Category的值为0x04。目前GON、P2P Invitation、Provision Discovery以及Device Discoverability使用P2P Public Action帧。

P2P Action帧监听事件：这种类型的帧属于802.11 Action帧的一种，其Category取值为0x7F，OUI指定为WFA的OUI（即50-6F-9A），而OUI-Type指定为P2P（取值为0x09）。目前Noticeof Absence、P2P Presence、GO Discoverability使用P2P Action帧。

-> wpas\_p2p\_init

//初始化一个p2p\_config对象

// drv\_flags

1. #define WPA\_DRIVER\_FLAGS\_AP        0x00000040
2. //Wifi driver支持AP。它使得P2P设备能扮演GO
3. #define WPA\_DRIVER\_FLAGS\_P2P\_CONCURRENT    0x00000200
4. //Wifi驱动支持STA和P2P的并发运行
5. #define WPA\_DRIVER\_FLAGS\_P2P\_CAPABLE    0x00000800
6. //Wifi驱动支持P2P

->p2p\_init

//初始化一个p2p\_data 的对象

###### P2P of wpa\_supplicant Device Discover

->CMD “ P2P\_FIND”

-> wpa\_supplicant\_ctrl\_iface\_process

//解析命令，调用p2p\_ctrl\_find

->p2p\_ctrl\_find

//解析cmd的参数内容，设置扫描内容

// P2P\_FIND\_START\_WITH\_FULL 先全信道扫描，再扫描social channel

// P2P\_FIND\_ONLY\_SOCIAL

// P2P\_FIND\_PROGRESSIVE 一直扫描所有的信道

-> wpas\_p2p\_find

// WPA\_DRIVER\_FLAGS\_P2P\_MGMT, 驱动注册中有此标志，调用驱动的p2p\_find,否则使用wpa\_supplicant的实现

->p2p\_find

//停止监听

//设置p2p的状态

//设置超时处理

->p2p\_scan->wpas\_p2p\_scan

//构建ie信息，wps p2p ssid

-> radio\_add\_work

//添加扫面任务到队列中，并设置回调函数wpas\_p2p\_trigger\_scan\_cb

-> radio\_start\_next\_work->wpas\_p2p\_trigger\_scan\_cb

//调用驱动发起扫描wpa\_drv\_scan

//回调函数中设置了扫描结果的处理函数wpas\_p2p\_scan\_res\_handler

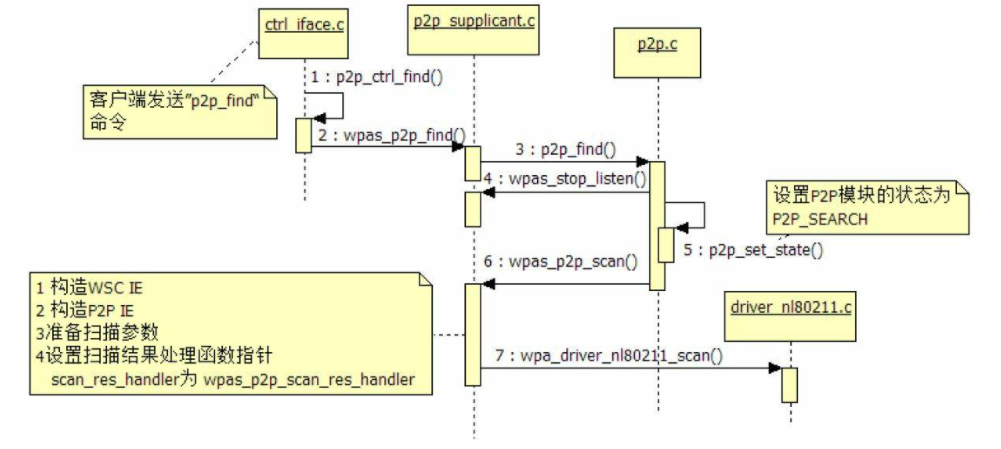
向驱动发起扫描：

->wpa\_drv\_scan->driver\_nl80211\_scan2

->wpa\_driver\_nl80211\_scan

//*打印log：nl80211: scan request*

//*NL80211\_CMD\_TRIGGER\_SCAN 发送命令->send\_and\_recv\_msgs*



UML顺序图

扫描结果处理：

->wpas\_p2p\_scan\_res\_handler

->p2p\_scan\_res\_handler //对每一个结果处理

->p2p\_add\_device //解析p2p wps两部分ie字段，存在msg中

// 过滤那些被阻止的P2P Device

// 构造一个p2p\_device对象，并将其加入p2p\_data结构体的devices链表中

//dev\_found函数指针指向wpas\_dev\_found，通过hidl向上层发送消息找到 P2P Device，该消息也称为P2P Device Found消息

->p2p\_scan\_res\_handled

->p2p\_run\_after\_scan//根据p2p\_find设置的标值决定下一个动作

->p2p\_continue\_find

->p2p\_listen\_in\_find //进入listen sate

//计算listen sate的时间

//组建rsp帧

->p2p->cfg->start\_listen=wpas\_start\_listen

//将p2p-listen加入到工作队列中radio\_add\_work

->p2p\_listen\_in\_find

//1.获取listen channel 2.计算停留时间 3.构造rsp帧

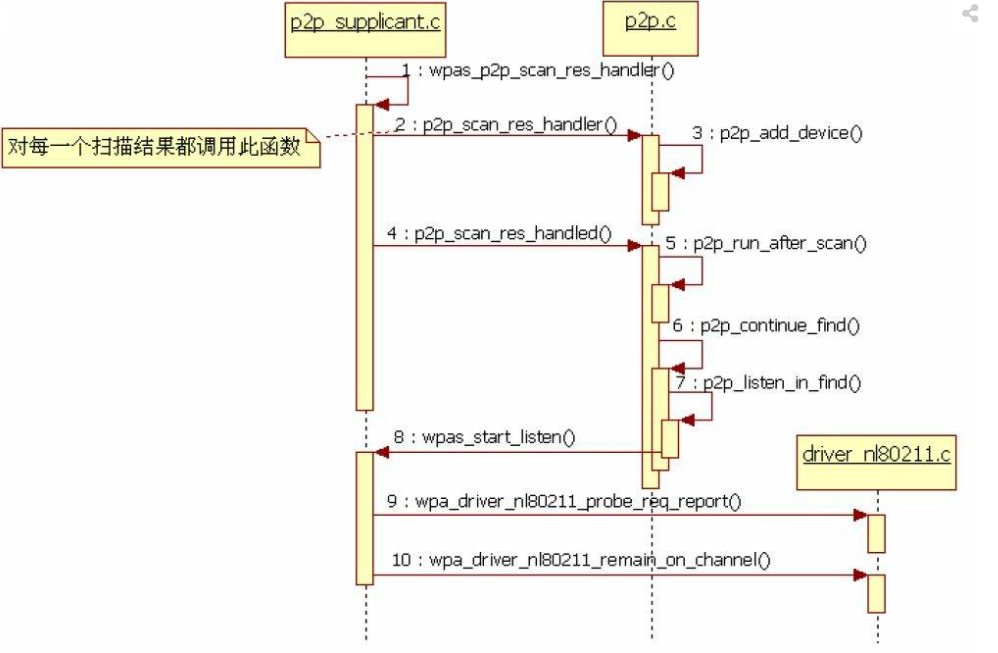
->start\_listen

->wpas\_start\_listen

->wpa\_drv\_set\_ap\_wps\_ie //android 特有接口

->wpa\_drv\_probe\_req\_report //收到req帧上报给wpa

->wpa\_drv\_remain\_on\_channel //让驱动再某个信道停留



###### P2P of wpa\_supplicant Provision Discover

1.上层命令：P2P\_PROV\_DISC xx:xx:xx:xx:xx pbc

-> p2p\_ctrl\_prov\_disc

//命令参数处理

-> wpas\_p2p\_prov\_disc //参数配置

-> p2p\_prov\_disc\_req

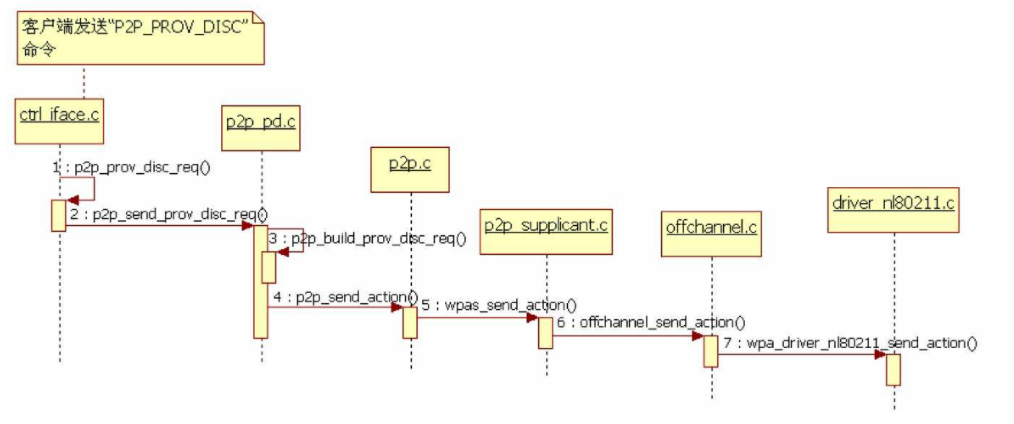
-> p2p\_get\_device//查找对象

-> p2p\_send\_prov\_disc\_req

//确定设备的工作频段

//构造req帧

->p2p\_send\_action



2.Action帧接收流程

-> process\_bss\_event(NL80211\_CMD\_FRAME)

->mlme\_event(NL80211\_CMD\_FRAME)

-> mlme\_event\_mgmt

-> wpa\_supplicant\_event

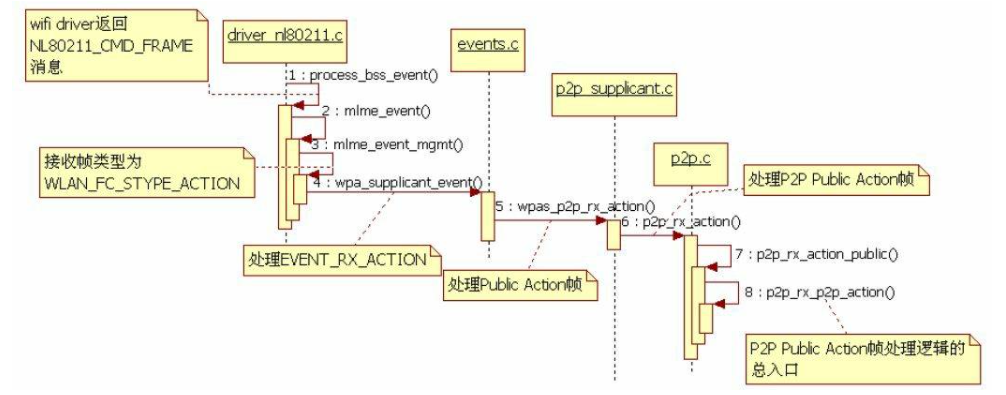
-> p2p\_rx\_action

-> p2p\_rx\_action\_public

-> p2p\_rx\_p2p\_action

//action帧分类处理的最后一个调用

//所有的p2p接受到的帧最终字此函数做分类处理



3.PD rsp帧的处理

-> p2p\_rx\_p2p\_action

-> p2p\_process\_prov\_disc\_resp

//判断wps字段设置的连接方式是否支持

-> wpas\_prov\_disc\_resp

-> wpa\_msg(P2P\_EVENT\_PROV\_DISC\_PBC\_RESP)

4.开始协商流程

上层应用收到P2P\_EVENT\_PROV\_DISC\_PBC\_RESP，对于Android而言P2pStateMachine会处理此消息

->P2P\_CONNECT (P2P\_CONNECT 8a：32：9b：6c：d1：80 pbc go\_intent=7)

->p2p\_ctrl\_connect //解析命令所带的参数

-> wpas\_p2p\_connect

//jion此变量协商完成为真否则为假

// wpas\_p2p\_create\_iface，若WPA\_DRIVER\_FLAGS\_P2P\_MGMT\_AND\_NON\_P2P标值存在则需要创建一个虚拟的interface供给p2p使用

-> wpas\_p2p\_start\_go\_neg

//设置超时时间，调用p2p\_connect

->p2p\_connect

//指定工作频段

//设置dev信息

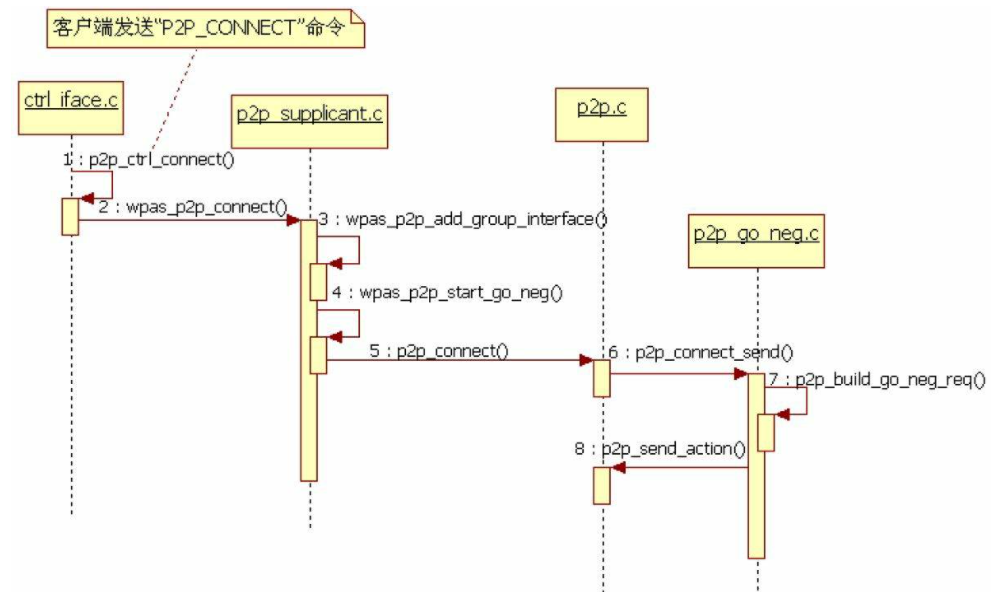
//停止find

->p2p\_connect\_send

//构建req帧

//设置p2p状态P2P\_CONNECT

->p2p\_send\_action //发送帧



5.GON RSP帧处理

-> p2p\_rx\_p2p\_action

-> p2p\_process\_go\_neg\_resp

//解析帧，决策go角色

//设置P2P状态P2P\_GO\_NEG

//构造nego comfirm帧

->p2p\_send\_action

//发送comfirm帧

发送comfirm帧，驱动将会发送一个NL80211\_CMD\_FRAME\_TX\_STATUS给内核，处理此消息会给wpa发送EVENT\_TX\_STATUS

6.EVENT\_TX\_STATUS处理流程

->offchannel\_send\_action\_tx\_status

->wpas\_p2p\_send\_action\_tx\_status

->p2p\_send\_action\_cb

//P2P\_PENDING\_GO\_NEG\_CONFIRM处理此分支

->p2p\_go\_neg\_conf\_cb

->wpas\_send\_action\_done

->p2p\_go\_complete

//设置状态P2P\_PROVISIONING

->wpas\_go\_neg\_completed

//wpa\_msg向上层发送P2P\_GO\_NEGOTIATION\_SUCCESS\_EVENT

//根据角色启动wps对应的流程egistrar/ Enrollee功能

