Tina Linux Wi-Fi 常见问题

调试指南

版本号: 1.5
发布日期: 2024.08.12

版本历史

版本号	日期	制/修订人	内容描述
1.0	2022.05.20	AWA1381	初始版本。
1.1	2022.12.01	AWA1381	1. 第 4 章, 添加排查思路章节。 2. 第 4 章, 添加一些案例分析。
1.2	2023.04.20	AWA1381	1. 修改部分错别字。 2. 添加部分代码的路径描述。
1.3	2023.10.23	AWA1381	1. 修改一些错误的标点符号使用。 2. 更改部分插图命名。
1.4	2023.12.12	AWA1436	1. 增加适用 buildroot 编译方式相关说明。
1.5	2024.08.12	KPA0568	1. 修改 tina5.0 下 buildroot 下已能使用 wifimanager。

目　录

插　图

1 前言

1.1 文档简介

本文档主要介绍 Tina Linux 平台上 Wi-Fi 常见问题排查手段，方便用户遇到 Wi-Fi 异常问题能通过该文档获取排查思路。

1.2 目标读者

- Wi-Fi 模块开发工程师。
- Wi-Fi 模块测试工程师。

1.3 适用范围

Tina Linux 所有平台。

1.4 文档约定

1.4.1 标志说明

⚠ 注意

- **提醒操作中应注意的事项。不当的操作可能会损坏器件，影响可靠性、降低性能等。**

说明

为准确理解文中指令、正确实施操作而提供的补充或强调信息。

技巧

一些容易忽视的小功能、技巧。了解这些功能或技巧能帮助解决特定问题或者节省操作时间。

1.4.2 地址与数据描述方法约定

本文档在描述地址、数据时遵循如下约定：

表 1-1: 地址与数据描述方法约定

符号	例子	说明
0x	0x0200，0x79	地址或数据以 16 进制表示。
0b	0b010，0b00 000 111	数据采用二进制表示 (寄存器描述除外)。
X	00X，XX1	数据描述中，X 代表 0 或 1。 例如，00X 代表 000 或 001；XX1 代表 001，011，101 或 111。

1.4.3 数值单位约定

本文档在描述数据容量（如 NAND 容量）时，单位词头代表的是 1024 的倍数；描述频率、数据速率等时则代表的是 1000 的倍数。具体如下：

表 1-2: 数值单位约定

类型	符号	对应数值
数据容量（如 NAND 容量）	1 K	1024
	1 M	1 048 576
	1 G	1 073 741 824
频率，数据速率等	1 k	1000
	1 M	1 000 000
	1 G	1 000 000 000

1.5 相关术语介绍

1.5.1 软件术语

表 1-3: 软件术语

术语	解释说明
RTLxxx	瑞昱驱动简写，意指所有瑞昱的无线模组, 例如：RTL8723DS,RTL8821CS。

术语	解释说明
XRADIO	全志驱动简写，意指所有全志的无线模组，例如：XR819S,XR829。
fw	firmware 的缩写，表示无线模组需要加载的一些固化代码。

2 常见调试开关

2.1 应用

2.1.1 wifimanager

(1) 源码修改

tina-dev/platform/allwinner/wireless/wifimanager/core/include/wifi_log.h

```
10 typedef enum __wmg_log_level{
11    WMG_MSG_NONE = 0,
12    WMG_MSG_ERROR,
13    WMG_MSG_WARNING,
14    WMG_MSG_INFO,
15    WMG_MSG_DEBUG,
16    WMG_MSG_MSGDUMP,
17    WMG_MSG_EXCESSIVE
18 }wmg_log_level_t;
```

tina-dev/platform/allwinner/wireless/wifimanager/core/src/log/wifi_log.c

```
27 static global_log_type_t global_log = {
28    .wmg_debug_level = WMG_MSG_INFO,
29    .wmg_log_para = 0,
30 };
......
78 void wmg_set_debug_level(wmg_log_level_t level)
79 {
80    global_log.wmg_debug_level = level;
81 }
82
83 int wmg_get_debug_level()
84 {
85    return global_log.wmg_debug_level;
86 }
```

(2) 命令修改

表 2-1: 打印等级修改命令

命令	解释说明
wifi -D exce	修改打印等级为 exce, 最高等级。
wifi -D dump	修改打印等级为 dump。
wifi -D debug	修改打印等级为 debug。
wifi -D info	修改打印等级为 info，默认打印等级。

命令	解释说明
wifi -D warn	修改打印等级为 warn。
wifi -D error	修改打印等级为 error。
wifi -D open	打开打印信息时间戳，文件路径，函数名，行号，默认不打开，与上面的打印等级不冲突。
wifi -D close	关闭打印信息时间戳，文件路径，函数名，行号。

例如：

```
root@TinaLinux:/# wifi -D open
root@TinaLinux:/# wifi -D warn
2023-08-07 11:43:21:564: WWAR: [src/os/linux/scan.c:parse_scan_results:191]scan ssid too long, we set to NULL
```

2.2 中间件

2.2.1 wpa_supplicant

说明

openwrt 编译方式支持该调试功能，tina5.0 下 buildroot 目前已能正常支持 wifimanager

(1) 源码修改

```
1.去到目录：tina/out/xxx[具体方案]/build_dir/target/hostapd-wpad-XXX/hostapd-2020-06-08-XXX/
1.1 git init;git add --all .;git commit -s(init: git init)
1.2 添加修改后git add .;git commit -s; git format-patch -1
1.3 将1.2生成的patch文件放到tina/openwrt/openwrt/package/network/services/hostapd/patches目录。
```

说明

该方式常用于需要自己添加打印的调试。

(2) 命令修改

```
1.package/network/services/hostapd/patches/410-limit_debug_messages.patch先删除了。
2.驱动wpa_supplicant时带-d参数，且在前台运行。
wpa_supplicant -i wlan0 -Dnl80211 -c/etc/wifi/wpa_supplicant.conf -O /etc/wifi/sockets -d  (-dd/-ddd)
```

说明

借助 wifi -D exce 直接设置最高等级，可以打开 wpa_supplicant 的默认打印。

2.3 驱动

驱动的调试控制涉及到具体厂商，这里主要介绍 xradio 和 realtek 的。

2.3.1 XRADIO

xradio 驱动默认添加 debugfs, 通过节点控制的方式改变各阶段的打印等级。如：

(1)/sys/kernel/debug/xradio_host_dbg 主要用于各流程的打印分析，如扫描，数据收发等。

```
root@TinaLinux:/sys/kernel/debug/xradio_host_dbg# ls
dbg_ap  dbg_logfile  dbg_sta  hwinfo  dbg_bh
dbg_pm  dbg_tpa_node set_sdio_clk  dbg_common  dbg_sbus
dbg_txrx  tx_burst_limit  dbg_etf  dbg_scan  dbg_wsm
```

例如：打开扫描时的打印：

```
root@TinaLinux:/sys/kernel/debug/xradio_host_dbg# echo 0xff >dbg_scan
root@TinaLinux:/sys/kernel/debug/xradio_host_dbg# wifi_scan_results_test
**********************************Start scan!**********************************
[ 139.041061] [SCAN] xradio_hw_scan
[ 139.047482] [SCAN] vif0 Scan request(2.4G-14chs) for 0 SSIDs.
[ 139.056753] [SCAN] xradio_remove_wps_p2p_ie
[ 139.070178] [SCAN] xradio_scan_work
[ 139.074120] [SCAN] Scan split ch(11).
[ 139.078486] [SCAN] xradio_scan_start
[ 140.255037] [SCAN] xradio_scan_complete_cb
[ 140.259684] [SCAN] xradio_scan_timeout
[ 140.263884] [SCAN] xradio_scan_complete
[ 140.268238] [SCAN] xradio_scan_work
[ 140.272149] [SCAN] Scan split ch(3).
[ 140.276191] [SCAN] xradio_scan_start
[ 140.623986] [SCAN] xradio_scan_complete_cb
[ 140.628648] [SCAN] xradio_scan_timeout
[ 140.632847] [SCAN] xradio_scan_complete
[ 140.637177] [SCAN] xradio_scan_work
[ 140.641105] [SCAN] Scan completed.
[ 140.644917] [SCAN] xradio_scan_restart_delayed
```

(2) 查看和设置 sdio 频率

```
cat /set_sdio_clk  默认是0表示扫卡时的频率。
echo 50000000 > set_sdio_ck  修改sdio频率为50M。
```

(3) 版本信息

```
root@TinaLinux:/sys/kernel/debug/ieee80211/phy0/xradio# cat version
Driver Label:XR_V02.16.84_P2P_HT40_01.31
Firmware Label:XR_C09.08.52.73_DBG_02.122 2GHZ HT40 May 18 2021 13:36:09
```

说明

也可以直接查看系统启动时驱动加载日志。

(4) 帧交互信息

echo 0xffff, 0xffff > /sys/kernel/debug/ieee80211/phy0/xradio/parse_flags

```
0x1 显示控制帧
0x2 显示管理帧（扫描相关帧除外）
0x4 显示数据帧
0x8 显示扫描相关帧
```

```
0x10 显示 TCP 协议帧
0x20 显示 UDP 协议帧
0x40 显示 DHCP 协议帧
0x80 显示 ICMP 协议帧（包含 ping 协议
0x100 显示 PF_8021X 协议帧
0x200 在信息中显示 MAC 层的序列号
0x400 在信息中显示自身 MAC 地址
0x800 在信息中显示源 MAC 地址和目标 MAC 地址
0x1000 在信息中显示无线层对端 MAC 地址
0x2000 在信息中显示 IP 地址
0x4000 显示未知协议的帧
```

例如：打开 dhcp 帧交互：

```
root@TinaLinux:/sys/kernel/debug/ieee80211/phy0/xradio# echo 0x40,0x40 > parse_flags
[ 545.289828] [XRADIO] txparse=0x0040, rxparse=0x0040
root@TinaLinux:/sys/kernel/debug/ieee80211/phy0/xradio# wifi_connect_ap_test AW-PDC-PD2-xiaomi2.4g 22224444
==================================
Connected to the AP(AW-PDC-PD2-xiaomi2.4g)
Getting ip address(AW-PDC-PD2-xiaomi2.4g)......
[ 572.295284] [XRADIO] if0-TX---DHCP, Opt=53, MsgType=1
[ 572.736012] [XRADIO] if0-RX---DHCP, Opt=53, MsgType=2
[ 572.743721] [TXRX_WRN] drop=1759, fctl=0x00d0.
[ 572.785287] [XRADIO] if0-TX---DHCP, Opt=53, MsgType=3
[ 572.801084] [XRADIO] if0-RX---DHCP, Opt=53, MsgType=5
Wifi connect ap : Success!
```

(5) 查看 xradio_bh,xradio_proc 的线程优先级

```
xr829/wlan/bh.c
xradio_bh()
{
   ret = sched_setscheduler(hw_priv->bh_thread, SCHED_FIFO, &param);
}
xradio_proc()
{
   ret = sched_setscheduler(hw_priv->proc.proc_thread,SCHED_FIFO, &param);
}
linux-5.4/include/uapi/linux/sched.h
 80 /*
 81  * Scheduling policies
 82  */
 83 #define SCHED_NORMAL      0
 84 #define SCHED_FIFO     1
 85 #define SCHED_RR       2
 86 #define SCHED_BATCH    3
 87 /* SCHED_ISO: reserved but not implemented yet */
 88 #define SCHED_IDLE     5
 89 #define SCHED_DEADLINE
```

2.3.2 RTLXXX

rtl 的调试控制在 makefile 中有个宏总体控制，打开就可以通过节点方式来控制打印等级。

以 rtl8723ds 为例：drivers/net/wireless/rtl8723ds/Makefile

(1) 基础调试等级

```
CONFIG_RTW_DEBUG = y //控制调试节点是否打开
CONFIG_RTW_LOG_LEVEL = 3 //可以设置默认调试等级[0-4]
echo 4 > /proc/net/rtl8723ds/log_level  //只要调试打开了，可以手动调整打印等级
```

(2) 查看 RTL8723DS 的 RTWHALXT, RTW_XMIT_THREAD,RTW_RECV_THREAD 的线程优先级。

```
rtl8723ds/core/rtw_xmit.c
rtw_xmit_thread()
{
   struct sched_param param = { .sched_priority = 1 };
   sched_setscheduler(current, SCHED_FIFO, &param);
}
rtl8723ds/core/rtw_recv.c
rtw_recv_thread()
{
   struct sched_param param = { .sched_priority = 1 };
   sched_setscheduler(current, SCHED_FIFO, &param);
}
rtl8723ds/hal/rtl8723d/sdio/rtl8723ds_xmit.c
rtl8723ds_xmit_thread()
{
   struct sched_param param = { .sched_priority = 1 };
   sched_setscheduler(current, SCHED_FIFO, &param);
}
linux-5.4/include/uapi/linux/sched.h
 80 /*
 81  * Scheduling policies
 82  */
 83 #define SCHED_NORMAL      0
 84 #define SCHED_FIFO     1
 85 #define SCHED_RR      2
 86 #define SCHED_BATCH    3
 87 /* SCHED_ISO: reserved but not implemented yet */
 88 #define SCHED_IDLE     5
 89 #define SCHED_DEADLINE     6
```

(3) 查看当前 sdio 频率

```
cat /proc/net/rtl8723ds/wlan0/sdio_card_info
```

2.4 SDIO

(1) 基础调试节点

```
cat /sys/devices/platform/soc/sdc1/mmc_host/mmc1/mmc1:0001/mmc1:0001:1/device   //设备id
root@Tina:/sys/devices/platform/soc/sdc1# ls
driver                sunxi_dump_clk_dly
driver_override          sunxi_dump_gpio_register
mmc_host              sunxi_dump_host_register
modalias              sunxi_host_filter_w_sector
of_node               sunxi_host_filter_w_speed
power                 sunxi_host_perf
subsystem               sunxi_insert
sunxi_dump_ccmu_register    uevent
```

重点介绍 sunxi_insert: 手动启动卡检测功能。其他节点是关于寄存器的操作，暂不需要了解。

```
echo 0 > /sys/devices/soc.0/1c0f000.sdmmc/sunxi_insert(告诉驱动卡被移除)
echo 1 > /sys/devices/soc.0/1c0f000.sdmmc/sunxi_insert(告诉驱动卡被插入)
```

3 排查思路

3.1 开发流程

无线开发流程如下：

图 3-1: 无线规范开发流程

说明

重点关注射频测试验证，分析问题前一定要确保硬件的 RF 指标是正常达标的，尤其是性能问题的分析。

3.2 分析思路

分析问题的思路按照：了解问题，确定问题，分析问题，反馈问题，总结问题的顺序进行。

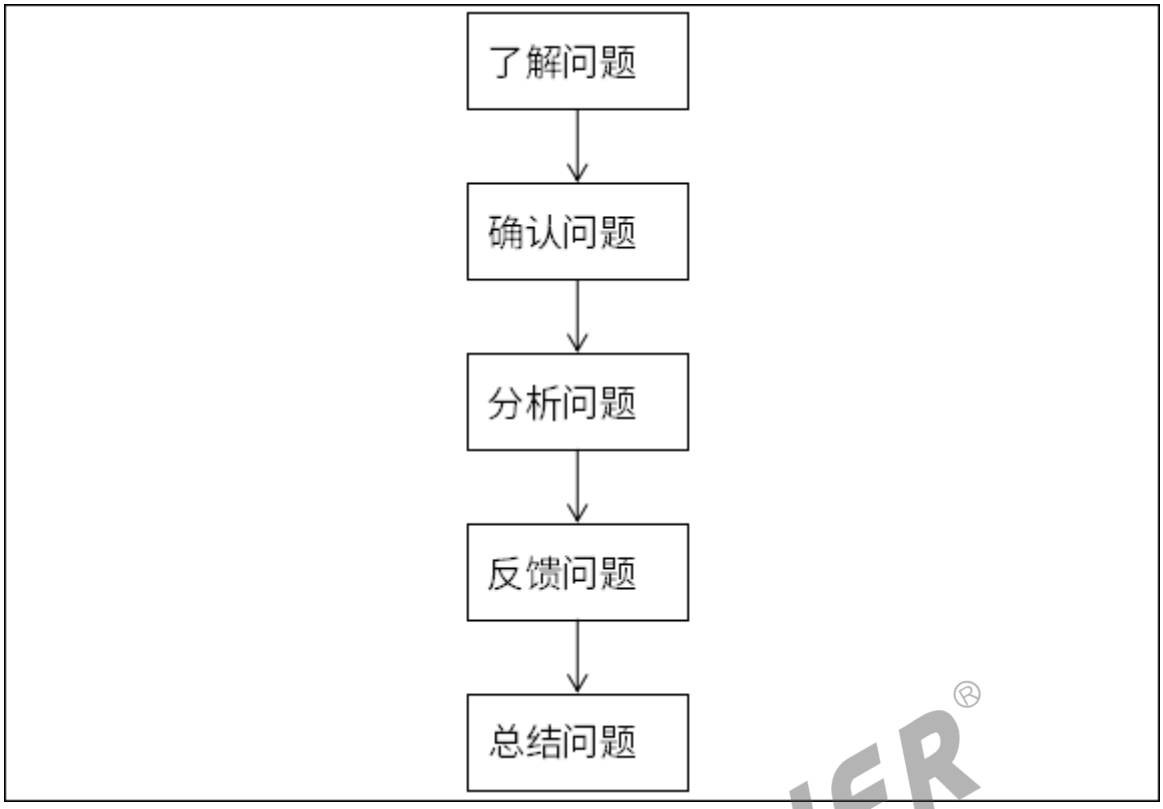

图 3-2: 问题分析思路

(1) 了解问题：需要了解问题的背景，所处环境，问题设备信息，关联设备信息。

图 3-3: 问题分析-了解问题

(2) 确认问题：需要进行硬件确认，软件确认，初步判断硬件单体问题，功能问题还是性能问题。

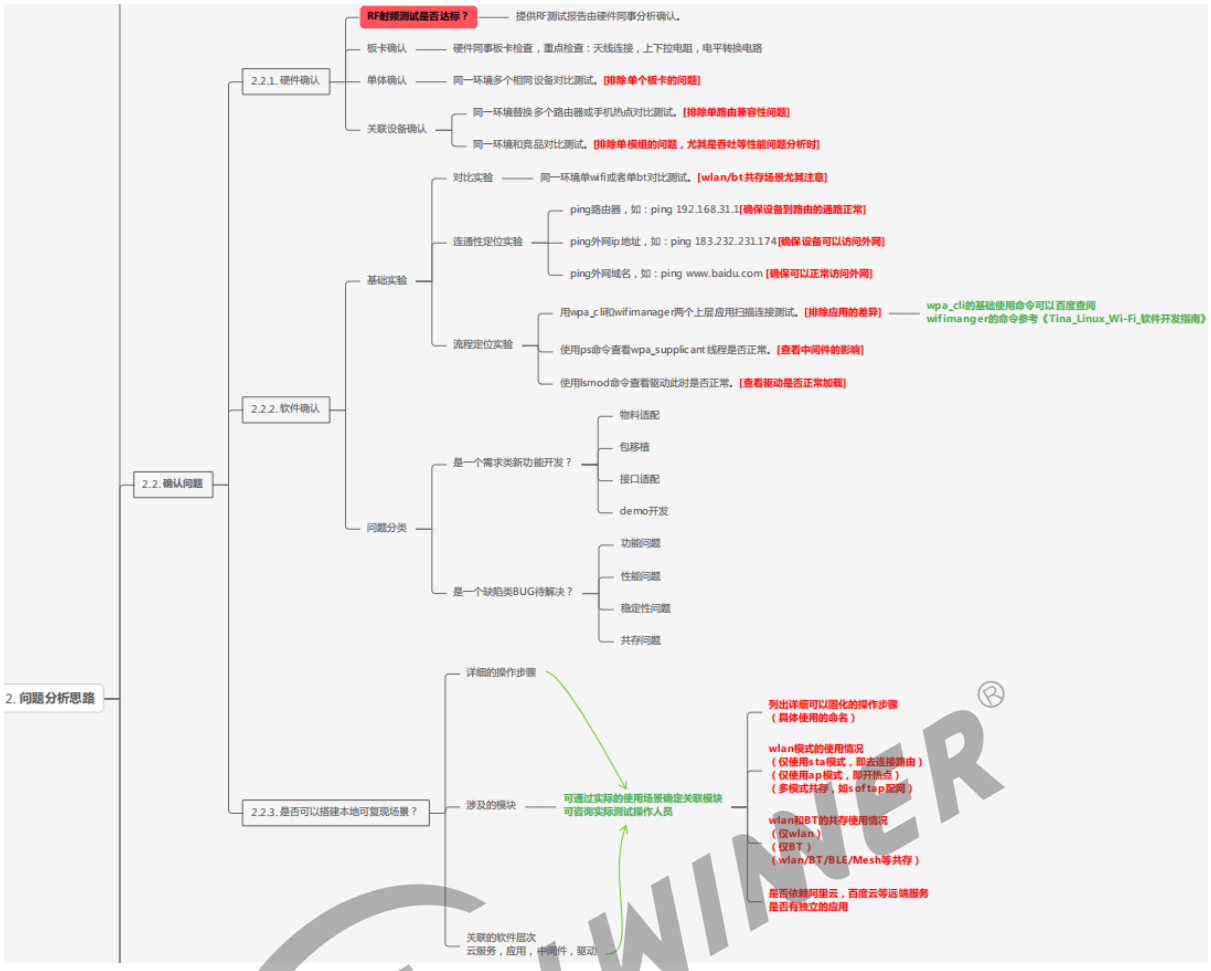

图 3-4: 问题分析-确认问题

(3) 分析问题：需要将问题分类，更详细的分析方法参考《Tina_Linux_Wi-Fi_ 常见问题排查指南》即本文档排查实例章节。

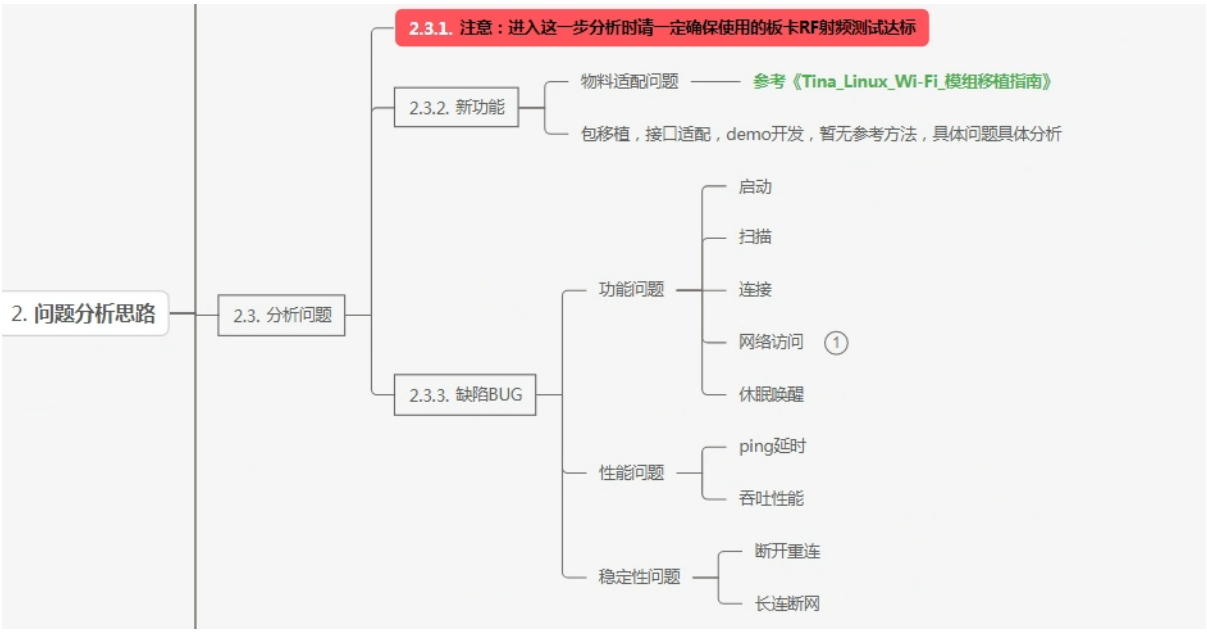

图 3-5: 问题分析-分析问题

(4) 反馈问题：根据排查指南分析仍没有结果的问题，避免来回沟通浪费时间，需要将已做实验进行详细的整理后再提。可以参考《Tina_Linux_Wi-Fi_ 模组移植指南》5.2 章节。

4 STA 模式

4.1 启动相关

4.1.1 排查思路

(1) 软件速查

1）执行：wifi_scan_results_test/wifi -s/wpa_cli进行扫描测试。详细命令参考《Tina_Linux_Wi-Fi_软件开发指南》。
2）执行：ifconfig wlan0查看网卡的状态。
3）执行：lsmod查看wlan模块加载的状态。
4）执行：ps | grep -rn wpa_supplicant | grep -v grep查看wpa_supplicant进程状态。
5）查看驱动加载日志，参考《Tina_Linux_Wi-Fi_模组移植指南》软件启动流程章节分析。
6）查看启动日志，参考《Tina_Linux_Wi-Fi_模组移植指南》软件启动流程章节分析。
7）确认firmware的版本，[重点注意24M/40M晶振的区别]。
8）确认dts中wlan属性的软件配置。参考《Tina_Linux_Wi-Fi_模组移植指南》硬件工作条件章节检查。

说明

步骤 1-8 没有绝对的顺序，优先做步骤 1-4 定位驱动，wpa_supplicant 服务，wlan0 起卡。

(2) 硬件排查

1）电压表或者示波器测量无线模块供电。
2）电压表或者示波器测量模块使能引脚电平。
3）示波器测量模块时钟波形。
4）检查上下拉电阻，尤其是特殊引脚（例如，有些模组的BT_WAKEUP_AP引脚被上拉会进入测试模式）。
5）尝试替换整颗物料。

说明

硬件部分可以请硬件同事协助分析，参考《Tina_Linux_Wi-Fi_ 模组移植指南》硬件工作条件章节检查。

4.1.2 案例一：fw 路径不匹配导致驱动加载失败

问题描述：R528+XR829 驱动加载失败。

问题背景：

产品：客户超市价签板卡。

主控：R528。

模组：XR829。

问题表现：lsmod 查看无 xr829。

问题分析：

1. 检查硬件是否正常。

- 供电：用万用表测量 VCC_WIFI, VCCIO_WIFI 供电正常。
- 使能：用示波器抓取 WLAN-REGON 引脚的波形，符合时序，且后面保持拉高。
- 时钟：用示波器抓取 SDIO_CLK，24M，32K 的信号波形图，输出正常。
- 外围：硬件同事审核外围器件的焊接，如电阻电容贴片，天线通路，以及模组正常。

2. 检查软件配置是否正常。

- 检查 dts 中关于 wlan 的引脚和功能配置正常。
- 检查 firmware 配置正常。

分析发现：firmware 配置和模组匹配，晶振选择匹配，但是路径和常用的不一致。

3. 分析系统启动日志。

系统启动时由 sunxi-rf 驱动负责配置的解析，在内核启动阶段一般可以看到如下类似打印：

```
[ 1.117385] sunxi-rfkill soc@3000000:rfkill@0: module version: v1.0.9
[ 1.139386] sunxi-rfkill soc@3000000:rfkill@0: wlan_busnum (1)
[ 1.145919] sunxi-rfkill soc@3000000:rfkill@0: Missing wlan_power.
[ 1.152844] sunxi-rfkill soc@3000000:rfkill@0: wlan clock[0] (32k-fanout1)
[ 1.160554] sunxi-rfkill soc@3000000:rfkill@0: wlan_regon gpio=204 assert=1
[ 1.168408] sunxi-rfkill soc@3000000:rfkill@0: wlan_hostwake gpio=202 assert=1
[ 1.176569] sunxi-rfkill soc@3000000:rfkill@0: wakeup source is enabled
```

4. 分析驱动加载日志

```
root@TinaLinux:/# insmod /lib/modules/5.4.61/xr829.ko
[16378.180038] ======== XRADIO WIFI OPEN ========
[16378.185389] [XRADIO] Driver Label:XR_V02.16.84_P2P_HT40_01.31
[16378.192185] [XRADIO] Allocated hw_priv @ 2e0a0149
[16378.199760] sunxi-rfkill soc@3000000:rfkill@0: bus_index: 1
[16378.216539] sunxi-rfkill soc@3000000:rfkill@0: wlan power on success
......
[16378.604372] mmc0: new high speed SDIO card at address 0001
[16378.611327] [SBUS] XRadio Device:sdio clk=50000000
[16378.618987] [XRADIO] XRADIO_HW_REV 1.0 detected.
[16378.674709] [XRADIO] xradio_update_dpllctrl: DPLL_CTRL Sync=0x00c00000.
[16378.708083] [XRADIO] Bootloader complete
[16378.712891] xradio_wlan mmc0:0001:1: Direct firmware load for fw_xr829.bin failed with error -2
[16378.722696] [XRADIO_ERR] xradio_firmware: can't load firmware file fw_xr829.bin.
[16378.731018] [XRADIO_ERR] xradio_load_firmware: can't download firmware.
[16378.738477] [XRADIO_ERR] xradio_load_firmware failed(-2).
[16378.744963] sunxi-rfkill soc@3000000:rfkill@0: wlan power off success
[16378.852274] [XRADIO] Remove SDIO card 1
......
failed to insert /lib/modules/5.4.61/xr829.ko
```

通过日志可以看到上电，扫卡都正常，但是在下载 firmware 时失败了。检查 firmware 的路径，命名。

根本原因：

firmware 文件路径异常，tina 上默认路径是/lib/firmware，客户自己把 firmware 文件的路径改为了/etc/firmware。

解决办法：

临时方案：将 fw_xr829.bin 文件正确命名推送到/lib/firmware 路径下即可。

最终方案：修改 firmware 的路径和 load 路径保持一致。

思路总结：

1. 检查硬件是否正常。

- 供电：和硬件工程师确定当前模组的供电方案，用万用表或者示波器实测模组的供电。
- 使能：根据模组规格书确定 WL-REG-ON 引脚的时序规则，用示波器抓取驱动加载过程 WL-REG-ON 的时序波形图。
- 时钟：用示波器抓取 SDIO_CLK，24M，32K 的信号波形图。
- 外围：硬件同事审核外围器件的焊接，如电阻电容贴片，天线通路，以及模组是否存在虚焊。

2. 检查软件配置是否正常。

- 检查 dts 中关于 wlan 的引脚和功能配置是否和硬件保持一致。
- 检查 firmware 配置。

3. 分析系统启动日志。

4. 分析驱动加载日志。

信息反馈：

按照排查指南仍无法解决，提供如下信息：

- 硬件原理图【含 PCB】。
- 提供软件配置【含 board.dts，config-4.9(5.4)，defconfig】。
- 提供初步分析的 sdio_clk, wlan-regon, 32k 的波形图。
- 提供系统启动完整日志。

4.1.3 案例二：wakeup 引脚未配置导致驱动加载失败

问题描述：Tina+R528+XR829 驱动加载异常导致 wifi 无法使用。

问题背景：

产品：标案。

主控：R528。

模组：XR829。

问题表现：

执行 wifi -o sta 失败。提示：

```
Successfully initialized wpa_supplicant
Could not read interface wlan0 flags: No such device
nl80211: Driver does not support authentication/association or connect commands
nl80211: deinit ifname=wlan0 disabled_11b_rates=0
Could not read interface wlan0 flags: No such device
wlan0: Failed to initialize driver interface
```

问题分析：

1. 复现问题

```
root@TinaLinux:/# wifi_daemon
...
root@TinaLinux:/# wifi -o sta
...
Successfully initialized wpa_supplicant
Could not read interface wlan0 flags: No such device
nl80211: Driver does not support authentication/association or connect commands
nl80211: deinit ifname=wlan0 disabled_11b_rates=0
Could not read interface wlan0 flags: No such device
wlan0: Failed to initialize driver interface
```

从提示可以看到 wlan0 异常，按一般思路确认驱动，wpa_supplicant，wlan0。

2. 确认驱动，wpa_supplicant,wlan0。

```
root@TinaLinux:/# ifconfig wlan0
ifconfig: wlan0: error fetching interface information: Device not found
root@TinaLinux:/# lsmod | grep -rn xr829
root@TinaLinux:/# ps | grep -rn wpa_supplicant | grep -v grep
```

因为系统起 wlan0 借助 wpa_supplicant 起服务，而 wpa_supplicant 起服务又借助驱动创建 wlan0 的节点。排查发现驱动加载失败了，优先分析驱动加载日志。

3. 手动加载驱动分析日志。

```
root@TinaLinux:/# insmod /lib/modules/5.4.61/xr829.ko
[  732.808177] ======== XRADIO WIFI OPEN ========
[  732.813819] [XRADIO] Driver Label:XR_V02.16.85_P2P_HT40_01.31
[  732.820727] [XRADIO] Allocated hw_priv @ dc9a08f9
[  732.827711] sunxi-rfkill soc@3000000:rfkill@0: bus_index: 1
[  732.844882] sunxi-rfkill soc@3000000:rfkill@0: wlan power on success
[  733.052151] sunxi-mmc 4021000.sdmmc: sdc set ios:clk 0Hz bm PP pm UP vdd 21 width 1 timing LEGACY(SDR12) dt B
[  733.052839] [XRADIO] Detect SDIO card 1
[  733.063428] sunxi-mmc 4021000.sdmmc: no vqmmc,Check if there is regulator
[  733.087786] sunxi-mmc 4021000.sdmmc: sdc set ios:clk 400000Hz bm PP pm ON vdd 21 width 1 timing LEGACY(SDR12)
        dt B
```

```
[ 733.112780] sunxi-mmc 4021000.sdmmc: sdc set ios:clk 400000Hz bm PP pm ON vdd 21 width 1 timing LEGACY(SDR12)
      dt B
[ 733.127496] sunxi-mmc 4021000.sdmmc: sdc set ios:clk 400000Hz bm PP pm ON vdd 21 width 1 timing LEGACY(SDR12)
      dt B
[ 733.149785] sunxi-mmc 4021000.sdmmc: sdc set ios:clk 400000Hz bm PP pm ON vdd 21 width 1 timing SD-HS(SDR25)
      dt B
[ 733.161429] sunxi-mmc 4021000.sdmmc: sdc set ios:clk 50000000Hz bm PP pm ON vdd 21 width 1 timing SD-HS(
      SDR25) dt B
[ 733.173344] sunxi-mmc 4021000.sdmmc: sdc set ios:clk 50000000Hz bm PP pm ON vdd 21 width 4 timing SD-HS(
      SDR25) dt B
[ 733.186008] mmc1: new high speed SDIO card at address 0001
[ 733.193385] [SBUS] XRadio Device:sdio clk=50000000
[ 733.208941] [XRADIO] XRADIO_HW_REV 1.0 detected.
[ 733.269401] [XRADIO] xradio_update_dpllctrl: DPLL_CTRL Sync=0x00c00000.
[ 733.303273] [XRADIO] Bootloader complete
[ 733.391111] [XRADIO] Firmware completed.
[ 733.395619] [XRADIO_ERR] xradio_request_gpio_irq: request_irq FAIL!ret=-22
[ 733.403388] [SBUS_ERR] sdio_irq_subscribe:xradio_request_gpio_irq failed(-22).
[ 733.411536] [XRADIO_ERR] xradio_load_firmware: can't register IRQ handler.
[ 733.419339] [XRADIO_ERR] xradio_load_firmware failed(-22).
[ 733.426547] sunxi-rfkill soc@3000000:rfkill@0: wlan power off success
[ 733.534670] sunxi-mmc 4021000.sdmmc: smc 1 p1 err, cmd 7, RTO !!
[ 733.541485] sunxi-mmc 4021000.sdmmc: smc 1 p1 err, cmd 7, RTO !!
[ 733.548286] sunxi-mmc 4021000.sdmmc: smc 1 p1 err, cmd 7, RTO !!
[ 733.555078] sunxi-mmc 4021000.sdmmc: smc 1 p1 err, cmd 7, RTO !!
[ 733.562316] mmc1: card 0001 removed
[ 733.562388] [XRADIO] Remove SDIO card 1
[ 733.566447] sunxi-mmc 4021000.sdmmc: sdc set ios:clk 0Hz bm PP pm OFF vdd 0 width 1 timing LEGACY(SDR12) dt B
[ 733.583736] sunxi-mmc 4021000.sdmmc: sdc set ios:clk 0Hz bm PP pm UP vdd 21 width 1 timing LEGACY(SDR12) dt B
[ 733.595009] sunxi-mmc 4021000.sdmmc: no vqmmc,Check if there is regulator
[ 733.610544] xradio_core_init failed (-22)!
[ 733.615281] sunxi-mmc 4021000.sdmmc: sdc set ios:clk 400000Hz bm PP pm ON vdd 21 width 1 timing LEGACY(SDR12)
      dt B
[ 733.639798] sunxi-mmc 4021000.sdmmc: smc 1 p1 err, cmd 52, RTO !!
[ 733.647510] sunxi-mmc 4021000.sdmmc: smc 1 p1 err, cmd 52, RTO !!
[ 733.654438] sunxi-mmc 4021000.sdmmc: sdc set ios:clk 400000Hz bm PP pm ON vdd 21 width 1 timing LEGACY(SDR12)
      dt B
[ 733.669165] sunxi-mmc 4021000.sdmmc: sdc set ios:clk 400000Hz bm PP pm ON vdd 21 width 1 timing LEGACY(SDR12)
      dt B
[ 733.682994] sunxi-mmc 4021000.sdmmc: smc 1 p1 err, cmd 5, RTO !!
[ 733.690623] sunxi-mmc 4021000.sdmmc: smc 1 p1 err, cmd 5, RTO !!
[ 733.698216] sunxi-mmc 4021000.sdmmc: smc 1 p1 err, cmd 5, RTO !!
[ 733.705971] sunxi-mmc 4021000.sdmmc: smc 1 p1 err, cmd 5, RTO !!
[ 733.712843] sunxi-mmc 4021000.sdmmc: sdc set ios:clk 0Hz bm PP pm OFF vdd 0 width 1 timing LEGACY(SDR12) dt B
failed to insert /lib/modules/5.4.61/xr829.ko
```

可以看到上电，扫卡的流程都正常，加载 firmware 异常，从日志可以看到异常原因：irq 请求失败。

```
[ 733.395619] [XRADIO_ERR] xradio_request_gpio_irq: request_irq FAIL!ret=-22
[ 733.403388] [SBUS_ERR] sdio_irq_subscribe:xradio_request_gpio_irq failed(-22).
[ 733.411536] [XRADIO_ERR] xradio_load_firmware: can't register IRQ handler.
[ 733.419339] [XRADIO_ERR] xradio_load_firmware failed(-22).
```

4. 分析启动日志。

```
[   0.815184] sunxi-rfkill soc@3000000:rfkill@0: module version: v1.0.9
[   0.822506] sunxi-rfkill soc@3000000:rfkill@0: get gpio chip_en failed
[   0.829864] sunxi-rfkill soc@3000000:rfkill@0: get gpio power_en failed
[   0.837303] sunxi-rfkill soc@3000000:rfkill@0: wlan_busnum (1)
```

```
[   0.843874] sunxi-rfkill soc@3000000:rfkill@0: Missing wlan_power.
[   0.850848] sunxi-rfkill soc@3000000:rfkill@0: wlan clock[0] (32k-fanout1)
[   0.858588] sunxi-rfkill soc@3000000:rfkill@0: wlan_regon gpio=145 assert=1
[   0.866613] sun8iw20-pinctrl pio: pio supply vcc-pe not found, using dummy regulator
[   0.875530] sunxi-rfkill soc@3000000:rfkill@0: get gpio wlan_hostwake failed
[   0.883830] sunxi-rfkill soc@3000000:rfkill@0: Missing bt_power.
[   0.890650] sunxi-rfkill soc@3000000:rfkill@0: bt clock[0] (32k-fanout1)
[   0.898200] sunxi-rfkill soc@3000000:rfkill@0: bt_rst gpio=210 assert=0
```

找到问题点，猜测是 board.dts 中 wlan_hostwake 引脚未配置，或者配置错误。

5.board.dts 确认 wlan 属性的软件配置。

根本原因：

board.dts 中 wlan_hostwake 引脚未配置。

解决办法：

增加 wlan_hostwake 配置。

```
wlan_hostwake  = <&pio PG 10 GPIO_ACTIVE_HIGH>;
```

思路总结：

1. 确认驱动，wpa_supplicant,wlan0。

2. 分析驱动日志

3. 分析启动日志。

4. 检查 dts 的 wlan 属性配置。

4.1.4 案例三：只读系统导致 wap_supplicant 启动失败

问题描述：R329+RTL8723DS wpa_supplicant 服务启动失败导致 wlan0 起卡失败。

问题背景：

产品：公板

主控：R329

模组：RTL8723DS

问题表现：ifconfig 找不到 wlan0。

问题分析：

1. 检查对应模组驱动是否正常加载。

lsmod 查看 rtl8723ds 驱动已经正常加载。

2. 检查 wpa_supplicant 应用是否正常启动。

wpa_supplicant 服务在 tina 系统默认是自启动的：/etc/init.d/wpa_supplicant

ps 查看线程

```
root@TinaLinux:/# ps | grep -rn wpa_supplicant
71: 1507 root
5524 S wpa_supplicant -iwlan0 -Dnl80211 -c/etc/wifi/wpa_sup
```

如果未正常启动，则手动执行：

```
root@TinaLinux:/# wpa_supplicant -i wlan0 -Dnl80211 -c/etc/wifi/wpa_supplicant.conf -O /etc
/wifi/sockets -B
Successfully initialized wpa_supplicant
[ 796.252780] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
mkdir[ctrl_interface=/etc/wifi/sockets]: Read-only file system
Failed to initialize control interface '/etc/wifi/sockets'.
```

从日志分析是因为文件系统只读，在起 wpa_supplicant 服务时无法从/etc/wifi 目录读取配置文件导致服务启动失败，最终导致起卡失败。

3. 手动起卡测试。

```
ifconfig wlan0 up
```

根本原因：

文件系统只读，起 wpa_supplicant 服务时无法从/etc/wifi 目录读取配置文件导致服务启动失败，最终导致起卡失败。

解决办法：

1. 修改/etc/wifi 目录的权限。
2. 将 wpa_supplicant.conf 文件放到其他可读写目录。

思路总结：

1. 检查对应模组驱动是否正常加载。

2. 检查 wpa_supplicant 应用是否正常启动。

3. 手动起卡测试

信息反馈：

按照排查指南仍无法解决，提供如下信息：

1. 明确驱动是否已经正常加载，系统启动完整日志。

2. 手动执行 lsmod , ifconfig wlan0 up 的日志。

3. 手动起 wpa_supplicant 服务的日志。【命令如上案例分析中】

4.2 扫描相关

4.2.1 排查思路

(1) 软件速查

1）执行：wifi_scan_results_test/wifi -s/wpa_cli进行扫描测试。详细命令参考《Tina_Linux_Wi-Fi_软件开发指南》。
2）执行：ifcofig wlan0查看网卡的状态。
3）执行：lsmod查看wlan模块加载的状态。
4）执行：ps | grep -rn wpa_supplicant | grep -v grep查看wpa_supplicant进程状态。

(2) 硬件排查

1）检查天线连接情况，板载天线的连接电阻，外置天线是否有接上。
2）检查sdio走线是否太差。
3）示波器测量晶振，检查起振情况。

说明

硬件部分可以请硬件同事协助分析，参考《Tina_Linux_Wi-Fi_ 模组移植指南》硬件工作条件章节检查。

4.2.2 案例一：fw 晶振版本使用错误导致无法扫描 ap

问题描述：MR813+XR829 扫描不到周围 AP。

问题背景：

产品：扫地机

主控：MR813

模组：XR829

问题表现：

系统起来，利用 wifimanager 进行扫描，无法扫描到任何周围 ap，也没有报错。打印如下：

```
root@TinaLinux:/# wifi_scan_results_test
**********************************
***Start scan!***
**********************************
bssid / frequency / signal level / flags / ssid
******************************
Wifi get_scan_results: Success!
******************************
```

问题分析：

1. 检查硬件天线为板载天线，且 RF 测试正常。

2.ifconfig 查看 wlan0 已经起卡。

3. 手动执行 wifimanager 扫描时没有报错。

4. 利用 wpa_cli 应用扫描同样也没有报错，但是也扫描不到周围 ap。

5. 同一个板卡替换 rtl8723ds 模组测试，可以正常扫描到。

说明还是 xr829 软件的问题，检查驱动，firmware, 发现是 firmware 文件用错了，模组贴的硬件物料是 40M 晶振，firmware 文件还是 24M 晶振的。

根本原因：

模组贴的硬件物料是 40M 晶振，firmware 文件还是 24M 晶振的。

解决办法：

拷贝 sdd_xr829_40M.bin 的 firmware 文件替换后正常。

思路总结：

1. 检查硬件天线是否焊接【尤其注意板载天线和外扣天线的差异】。

2. 检查网卡是否成功起卡。

3. 利用 wifimanager 扫描时是否有明显报错。

4. 利用 wpa_cli 应用扫描排查上层应用问题。

5. 模组替换对比测试排查 wpa_supplicant 和驱动差异。

6. 抓空中包分析。

信息反馈：

按照排查指南仍无法解决，给出当前初步分析定位的点，提供如下信息：

- 定位是 wifimanager 的问题，提供扫描的日志
- 定位是 wpa_supplicant 的问题，提供软硬件环境和复现方法描述。
- 定位是驱动源码问题，提供软硬件环境和复现方法描述。

问题描述：Tina+R528+XR829 sdd 文件晶振用错导致 wifi 无法使用。

问题背景：

产品：标案

主控：R528

模组：XR829

问题表现：使用 wifi -c 发现无法联网。

问题分析：

(1) 复现问题

```
wifi_daemon
wifi -o sta
wifi -c allwinner-wpa2 Aa123456
2022-11-28 13:24:05:605: WMG_ERROR [wifi_daemon.c:cmd_handle_c:673]:  ===Wi-Fi connect failed,time 0.000000 ms
        ===
```

(2) 扫描测试

```
wifi -s
2022-11-28 13:37:58:755: WMG_WARNG [wifi_daemon.c:cmd_handle_s:595]:  ===Wi-Fi scan failed, time 1180.000000 ms
        ===
```

发现都无法扫描到周围 ap，进一步先确认驱动,wpa_supplicant，wlan0，天线。

(3) 确认驱动,wpa_supplicant，wlan0。

```
root@TinaLinux:/# lsmod | grep -rn xr829
5:xr829              454656  0
root@TinaLinux:/# ps | grep -rn wpa_supplicant | grep -v grep
61: 1825 root     2440 S    wpa_supplicant -iwlan0 -Dnl80211 -c/etc/wifi/wpa_sup
root@TinaLinux:/# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr A8:3A:D7:53:79:61
        UP BROADCAST MULTICAST  MTU:1500  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

(4) 天线为板载天线，确认正常。

(5) 确认模组晶振和 sdd 文件的版本是否匹配。

根本原因：模组为 40M 晶振，软件配置的是 24M 晶振的 sdd_xr829.bin 文件。

解决办法：

替换为 sdd_xr829_40M.bin 文件重命名为：sdd_xr829.bin

openwrt 编译系统：执行 make menuconfig 配置时选择 40M 的 module

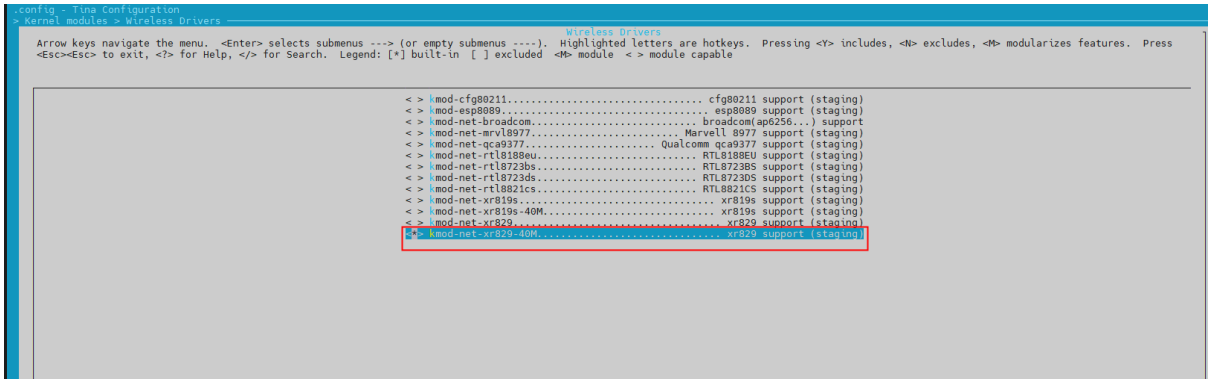

图 4-1: xr829_40M_tina 配置

buildroot 编译系统：执行 build.sh buildroot_menuconfig 配置时选择 40M 的 module

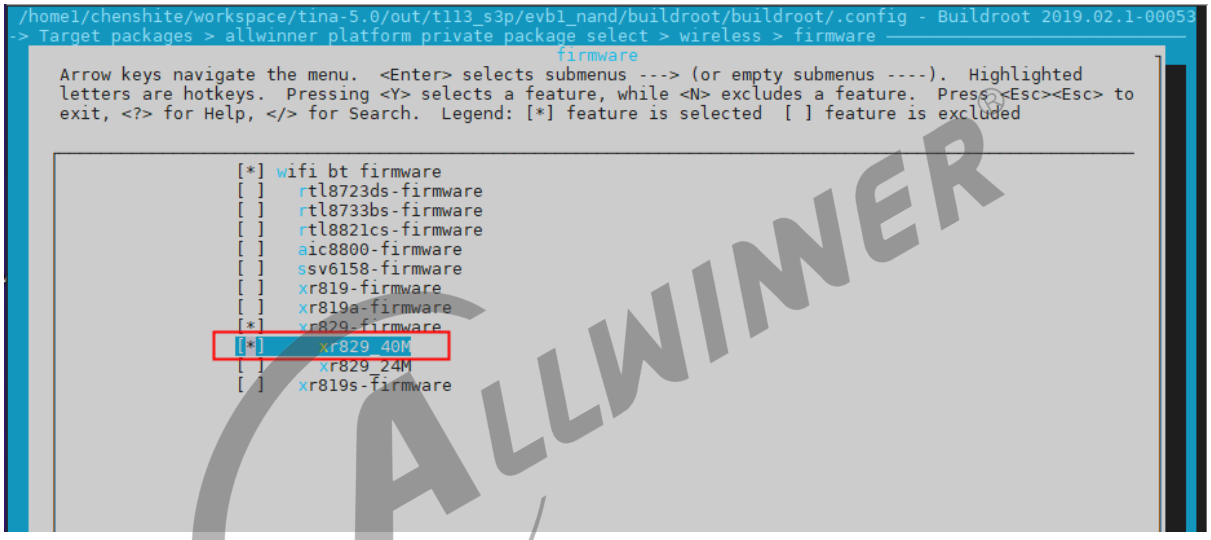

图 4-2: buildroot_xr829_40M_tina 配置

思路总结：

1. 先做扫描测试。

2. 确认驱动，wpa_supplicant,wlan0 的软件情况。

3. 确认硬件天线。

4. 确认 firmware 版本，尤其时晶振匹配。

5. 再针对板卡做完整的硬件检查。

4.2.3 案例二：扫描的 ap 数量少

问题描述：R328+xr829 用 wifimanager 只能扫描到 2 个 AP。

问题背景：

产品：公板

主控：R328

模组：XR829

问题表现：

系统起来 wifimanager 扫描，扫描到周围的 ap 特别少，有时候只有一两个。打印如下：

```
root@TinaLinux:/# wifi_scan_results_test
**********************************
***Start scan!***
**********************************
bssid / frequency / signal level / flags / ssid
50:d2:f5:f1:b7:08 2462 -25 [WPA-PSK-CCMP+TKIP][WPS][ESS] AW-PDC-PD2-xiaomi2.4g
5c:a4:8a:bf:25:72 2437 -57 [WPA2-PSK-CCMP][ESS] AWTest
*******************************
Wifi get_scan_results: Success!
*******************************
```

问题分析：

扫描周围 ap 与信号强度密切相关，能够扫描到，说明基本功能正常，大概率是天线的问题，天线分为板载的和外置的。

硬件排查发现板载天线通路异常，电阻未焊接。

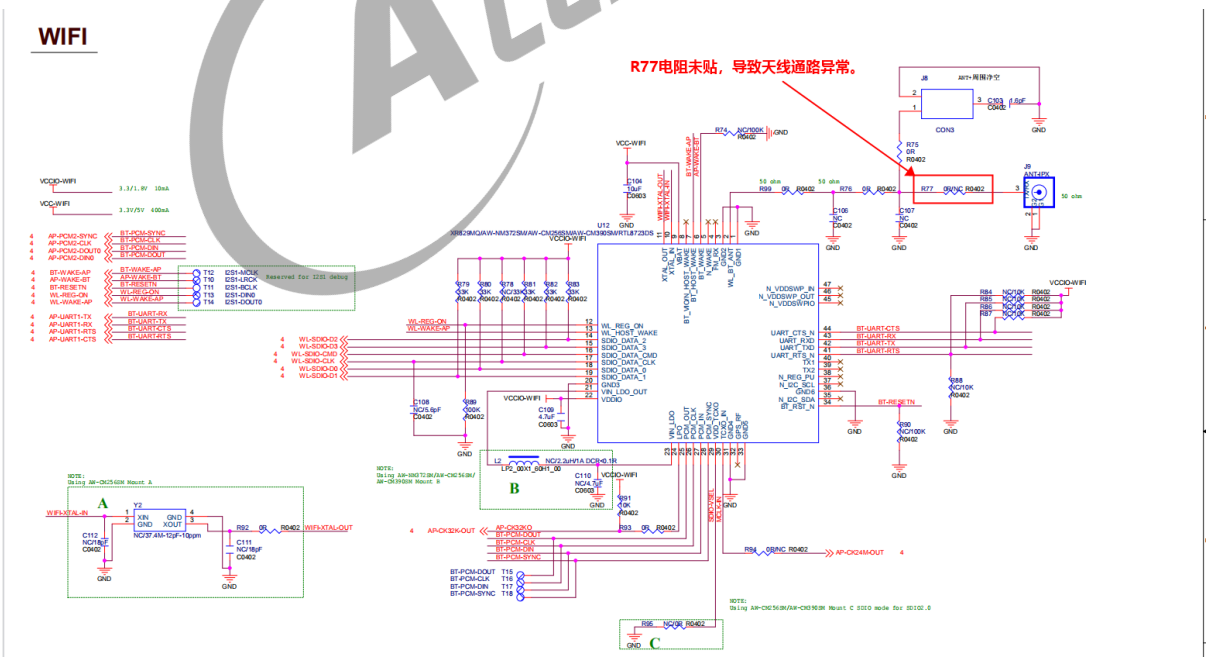

图 4-3: 扫描 ap 少天线原理图

根本原因：板载天线硬件通路电阻未焊接。

解决办法：如图硬件补贴 0 欧电阻 R77。

思路总结：

1. 检查硬件是否有异常。

- 板载天线是否焊接通路电阻，阻抗匹配是否正常。
- 直接焊接外置天线测试。

2. 环境对比试验。

- 同环境下，多个设备扫描对比测试，排除个例因素。
- 同环境下，有条件的话直接同板卡同系统，换个模组进行扫描测试。
- 干净环境下，有条件的话屏蔽房进行扫描测试。

3. 稳定场景具体分析。

- wpa_cli 进行扫描测试，排除 wifimanager 应用问题。
- wpa_supplicant 打印分析扫描结果。
- 驱动打印分析扫描结构，必要时抓包分析。

信息反馈：

按照排查指南仍无法解决，给出当前初步分析定位的点，提供如下信息：

- 反馈环境对比测试的结论，确认是板卡共性问题。
- 稳定复现的场景，提供软硬件方便复现排查。

4.3 连接相关

4.3.1 排查思路

(1) 软件速查

```
1）执行：wifi_scan_results_test/wifi -s/wpa_cli进行扫描测试。详细命令参考《Tina_Linux_Wi-Fi_软件开发指南》。
注：如果不能正常扫描到周围ap，请参考4.2章节扫描相关问题分析。
2）进一步确认是否可以扫描到连接的ap以及该ap的信号强度。
3）确认firmware的版本。[重点注意24M/40M晶振的区别]
4）使用wifi_scan_results_test/wifi -s/wpa_cli进行连接测试。详细命令参考《Tina_Linux_Wi-Fi_软件开发指南》。
5）检查ap的属性设置
  加密方式：open/wpa/wpa2/wpa3/wep
  频段：2.4G/5G
  带宽：HT20/HT40
```

信道：1~13
Ssid:名称是否含中文等特殊字符
Ap：兼容性

说明

关于第 4 点需要针对 ap 做对比测试，比如更换加密方式连接测试，同一个环境替换路由连接测试等。

(2) 硬件排查

1）检查天线连接情况，板载天线的连接电阻，外置天线是否有接上。
2）检查sdio走线是否太差。
3）示波器测量晶振，检查起振情况。
4）替换一个板卡测试。

说明

硬件部分可以请硬件同事协助分析，参考《Tina_Linux_Wi-Fi_ 模组移植指南》硬件工作条件章节检查。

(3) 软件细查

经过前面的分析基本可以缩小到：应用，wpa_supplicant，驱动，ap 兼容性问题。

1）打开wifimanager,wpa_supplicant,驱动（连接过程）的打印，连接测试抓日志分析，参考[第2章节]。
2）连接过程抓包分析，参考《Tina_Linux_Wi-Fi_抓包使用指南》。

4.3.2 案例一：mac80211 未配置导致非加密 ap 连接失败

问题描述：H133+XR829 模组无法连接开放性 ap。

问题背景：

产品：公板

主控：H133

模组：XR829

问题表现：系统起来使用 wifimanager 无法连接开放性 ap，进一步测试也无法连接加密 ap。

问题分析：

利用 wifimanager 进行扫描测试可以正常扫描到周围 ap。

在屏蔽房测试连接，同样连接失败。

所有板卡表现一致。

利用 wpa_cli 工具测试同样无法连接，排除应用影响。

同一版本的驱动在另外的主控 R328+XR829 板子上测试正常，排除驱动和 firmware 的影响。

两个方案就只剩内核配置的差异了，对比发现 h133 未配置 mac80211。

根本原因：

h133 未配置 mac80211。

解决办法：

内核配置 mac80211

```
> Networking support > Wireless
<*> Generic IEEE 802.11 Networking Stack (mac80211)
```

思路总结：

1. 扫描测试

进行扫描测试确认基本功能，如果扫描正常，说明板卡硬件正常。无法正常扫描，请参考 3.2 节扫描问题分析。

2. 对比测试

- 同环境下，其他同样的板卡对比测试，排除单个板卡问题。
- 同板卡，干净环境乃至屏蔽房进行非加密测试，排除环境干扰。
- 同环境下，切换路由器设备或者手机开热点测试，排除设备 ap 兼容性问题。

3. 定位分析

基于第一步测试后，基本可以定位该模组在当前系统下肯定是存在问题的，这里讨论的都是基于 Tina 系统的 wifimanager 测试的，一般关注：

内核版本，wpa_supplicant，驱动。

快速定位驱动：同板卡，同系统，替换一个模组来做对比测试。

快速定位内核：同板卡，同驱动，不同内核版本来做对比测试。

wpa_supplicant：Tina 系统的 wpa_supplicant 版本没有随着系统版本升级而升级，目前看来

出问题概率很小，所以待上面所有测试完成后最后再来分析。

4. 根因分析

- 若经过第二步定位是驱动问题：
- 打开驱动调试。【参见 2.0 章节】
- 抓空包分析。
- 若经过第二步定位是内核问题：
- 检查内核配置，重点关注：CONFIG_CFG80211，CONFIG_MAC80211，CRYPTO

信息反馈：

按照排查指南仍无法解决，给出当前初步分析定位的点，提供如下信息：

- 反馈测试后定位的点，是驱动，内核，wpa_supplicant 的问题。
- 如果定位是内核版本问题，提供内核配置问题，config-4.9/config-5.4 等。
- 如果定位是驱动版本问题，提供添加打印的测试日志。
- 如果定位是 wpa_supplicant 的问题，提供添加打印的测试日志。
- 无法定位具体点话，提供软硬件环境，便于复测分析。

4.3.3 案例二：连接 WEP 加密失败

问题描述：R528+XR819s，路由设置 WEP 加密方式，连接失败。

问题背景：

产品：公板

主控：R528

模组：XR819s

问题表现：

连接加密 AP 失败，提示：

```
root@TinaLinux:/# wifi_connect_ap_test AW-PDC-PD2-wep 0123456789
==================================
Connecting to the network(AW-PDC-PD2-wep).....
Connecting to the network(AW-PDC-PD2-wep).....
SET_NETWORK 0 wep_key0 0123456789 failed,Remove the information just connected!
Disconnected,the reason:WSE_CMD_OR_PARAMS_ERROR
Wifi connect ap : Failure!
```

问题分析：

利用 wifimanager 可以正常扫描到周围 ap，也包括要连接的 ap。

设置非加密的 ap 可以正常连接。

分析加密方式，当前设置的是 WEP 加密，切到 WPA/WPA2 加密方式后可以正常连接。

自此问题转化为：无法连接 WEP 加密方式的 ap。

进行对比测试，同环境，同板卡，仅替换模组测试，RTL8723DS 可以正常连接。

问题进一步转化为：XR819s 驱动在 linux-5.4 内核下不支持 WEP 加密方式连接。

进行定位分析：

打开驱动分析，对比 WPA 和 WEP 两种加密方式的连接日志发现 WPE 的没有发起认证。

```
echo 0xff > /sys/kernel/debug/xradio_host_dbg/dbg_sta
```

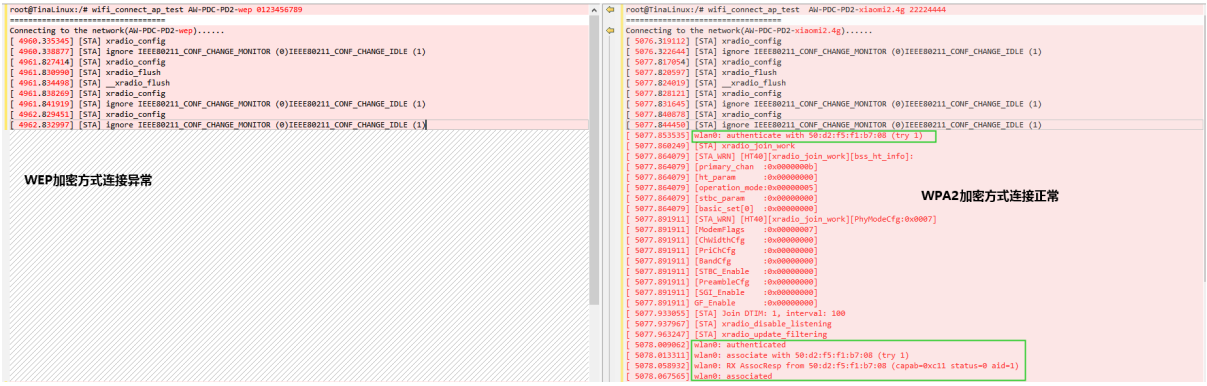

图 4-4: WEP 加密连接失败对比 wpa2

进一步打开驱动的 txrx 打印：

```
echo 0xff > /sys/kernel/debug/xradio_host_dbg/dbg_txrx
```

日志分析：

```
root@TinaLinux:/# wifi_connect_ap_test AW-PDC-PD2-wep 0123456789
==================================
Connecting to the network(AW-PDC-PD2-wep).....
[ 60.300508] [TXRX] xradio_rx_cb
[ 60.304048] [TXRX] xradio_frame_monitor
[ 60.312080] [TXRX] xradio_rx_cb
[ 60.315673] [TXRX] xradio_frame_monitor
[ 60.320065] [TXRX] xradio_rx_cb
[ 60.323691] [TXRX] xradio_frame_monitor
[ 60.328048] [TXRX] xradio_rx_cb
[ 60.331629] [TXRX] xradio_frame_monitor
[ 60.336008] [TXRX] xradio_rx_cb
.....
```

反复打印，从这里基本可以判断是 tx/rx 流程异常了。

分析源码最后发现：tx 流程多次分配 IV 导致对方无法正常解析，rx 流程没有正确去除 ICV，导致在 umac 层丢包。

根本原因：tx 流程多次分配 IV 导致对方无法正常解析，rx 流程没有正确去除 ICV，导致在 umac 层丢包。

解决办法：

在 xr829/wlan/sta.c-》xradio_set_key() 中将 IEEE80211_KEY_FLAG_ALLOC_IV 标志去除，以避免多次分配 IV。

在 xr829/umac/wep.c-》mac80211_crypto_wep_decrypt() 中修改 if 条件判断，确保 skb 能上传。

```
diff --git a/drivers/net/wireless/xr819s/umac/wep.c b/drivers/net/wireless/xr819s/umac/wep. c
index d9aff544216d..94e7e7e8ab54 100644
--- a/drivers/net/wireless/xr819s/umac/wep.c
+++ b/drivers/net/wireless/xr819s/umac/wep.c
@@ -276,7 +276,10 @@ mac80211_crypto_wep_decrypt(struct ieee80211_rx_data *rx)
return RX_DROP_UNUSABLE;
ieee80211_wep_remove_iv(rx->local, rx->skb, rx->key);
/* remove ICV */
-       if (!pskb_trim(rx->skb, rx->skb->len - IEEE80211_WEP_ICV_LEN))
+       /* NOTE: !(status->flag & RX_FLAG_ICV_STRIPPED) maybe used for
+        * controlompatibility with other versions of umac.
+        */
+
if (pskb_trim(rx->skb, rx->skb->len - IEEE80211_WEP_ICV_LEN))
return RX_DROP_UNUSABLE;
}
diff --git a/drivers/net/wireless/xr819s/wlan/sta.c b/drivers/net/wireless/xr819s/wlan/sta.c
index dda229086f40..bfb9981ede1d 100644
--- a/drivers/net/wireless/xr819s/wlan/sta.c
+++ b/drivers/net/wireless/xr819s/wlan/sta.c
@@ -1042,8 +1042,6 @@ int xradio_set_key(struct ieee80211_hw *dev, enum set_key_cmd cmd,
if (sta)
peer_addr = sta->addr;
-       key->flags |= IEEE80211_KEY_FLAG_ALLOC_IV;
-
priv->cipherType = key->cipher;
switch (key->cipher) {
case WLAN_CIPHER_SUITE_WEP40:
```

思路总结：

1. 进行扫描测试

进行扫描测试确认基本功能，如果扫描正常，说明板卡硬件正常。无法正常扫描，请参考 3.2 章节扫描问题分析。

2. 非加密 AP 连接测试

进行非加密 AP 连接测试，分析连接过程：

认证: 搜索打印关键字【auth】

关联: 搜索打印关键字【assoc】

获取 ip:ifconfig 查看 ip

若非加密也无法连接，参考 3.3 节案例一。

3. 定位分析

定位加密方式：切换加密方式测试，分析是具体某一种加密不支持，还是所有加密方式都不支持。

定位兼容性：切换 ap 进行测试，定位 ap 设备兼容性问题。

定位驱动：同环境下，同板卡仅替换模组测试。

定位内核版本：同板卡，同驱动，不通内核版本测试。

4. 根因分析

a. 若定位是驱动问题：

- 打开驱动调试。【参见 2.0 章节】
- 抓空包分析。

b. 若定位是内核问题：

- 检查内核配置，重点关注：CONFIG_CFG80211，CONFIG_MAC80211，CRYPTO。

信息反馈：

按照排查指南仍无法解决，给出当前初步分析定位的点，提供如下信息：

- 反馈初步测试和分析，是兼容性还是共性问题。
- 对比测试定位是驱动还是内核问题，甚至定位具体的加密方式。
- 驱动问题提供日志，联系原厂分析。
- 提供软硬件环境，便于复测排查。

4.3.4 案例三：联网失败

问题描述：Tina+R528+XR829 联网失败

问题背景：

产品：标案

主控：R528

模组：XR829

问题表现：执行 wifi -c ssid passwd 联网失败。

问题分析：

1. 复现问题

```
2022-11-28 15:44:20:581:WMG_INFO[wifi_daemon.c:cmd_handle_c:651]:===Wi-Fi connect use wpa2 failed ,try wep
2022-11-28 15:44:27:439:WMG_INFO[wifi_daemon.c:cmd_handle_c:661]:===Wi-Fi connect use wpa2/wep failed ,try wp3
2022-11-28 15:44:27:443:WMG_ERROR[src/wmg_sta.c:sta_connect:170]:  failed to connect ap
2022-11-28 15:44:27:443:WMG_ERROR[src/wmg_common.c:__wifi_sta_connect:191]:  wifi station connect fail
2022-11-28 15:44:27:443:WMG_ERROR[wifi_daemon.c:cmd_handle_c:673]:  ===Wi-Fi connect failed,time 0.000000 ms
```

日志分析看到联网的确失败了，尝试了三次都失败, 分别是 wpa/wpa2,wep,wpa3 的加密方式。

2. 执行扫描测试

```
root@TinaLinux:/# wifi -s
2022-11-28 15:48:16:258: WMG_INFO [wifi_daemon.c:cmd_handle_s:590]:  bss[18]: bssid=b0:48:7a:51:83:82  ssid=
        allwinner-wep  channel=1(freq=2412)  rssi=-35  sec=NONE
2022-11-28 15:48:16:590: WMG_INFO [wifi_daemon.c:cmd_handle_s:592]:  ===Wi-Fi scan successful, total 22 ap(buff size:
         60) time 1490.000000 ms===
```

显然可以扫描到 ap，而且信号也很好。说明驱动，wpa_supplicant，wlan0 启动都正常。加下来分析 ap 的属性设置。

3. 发现 ap 采用了 wep 的加密方式。替换一个 wpa/wpa2 的路由测试。

```
root@TinaLinux:/# wifi -c allwinner-wpa2 Aa123456
2022-11-28 15:50:58:300: WMG_INFO [wifi_daemon.c:cmd_handle_c:646]:  ===Wi-Fi connect use sec(3)===
2022-11-28 15:50:58:300: WMG_INFO [wifi_daemon.c:cmd_handle_c:647]:  ===Wi-Fi connect successful,time
        7090.000000 ms===
```

连接成功，基本可以判定该问题是加密方式属性的差异。

4. 需要重新定位是 wifimanager,wpa_supplicant, 还是驱动的问题。采用控制变量法。

4.1wpa_cli 工具测试，现象一致，wpa/wpa2 加密的 ap 可以正常连接，wep 的 ap 无法连接。

- 排除 wifimanager 的影响。

4.2wpa_supplicant 用的是开源包，一般问题不大，优先怀疑驱动。

4.3 替换一个 rtl8723ds 模组测试，wpa/wpa2/wep 都连接正常。

- 从这个实验基本可以定位是驱动的影响了。

5. 根因分析，打开 xr829 驱动的日志，同时抓包分析。

从前面的分析基本可以判定就是 xr829 驱动不支持 wep 加密方式，所有为了方便分析，我们把 wifimanager 的日志关闭，把驱动 sta 模式的日志打开。

```
root@TinaLinux:/# wifi -D error
root@TinaLinux:/# echo 0xff > /sys/kernel/debug/xradio_host_dbg/dbg_sta
```

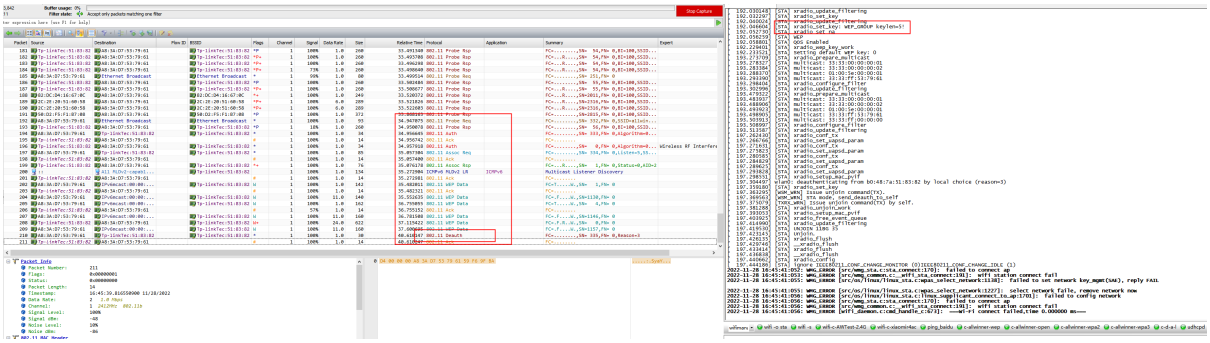

图 4-5: 4.3.4-1 抓包 wep 连接失败

从抓包分析看就是认证失败后设备自己发了 deauth 就断开了。很难有多余的信息。一般这种问题到这里从正面分析，如果对流程不熟悉，就比较难了，所有我们采用对比法。找一个正常可以连接 wep 的板子做对比。因为之前已经支持过，我这里将驱动版本回退后优先抓日志对比。

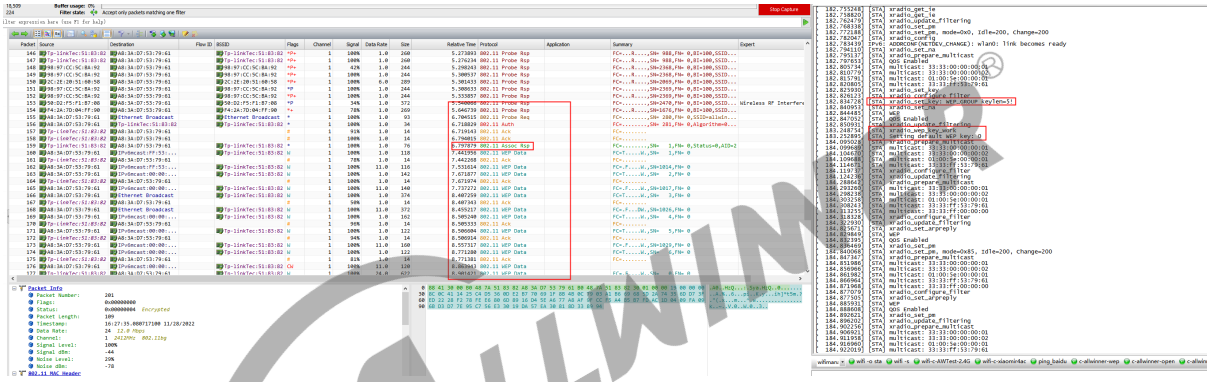

图 4-6: 4.3.4-2 抓包 wep 连接成功

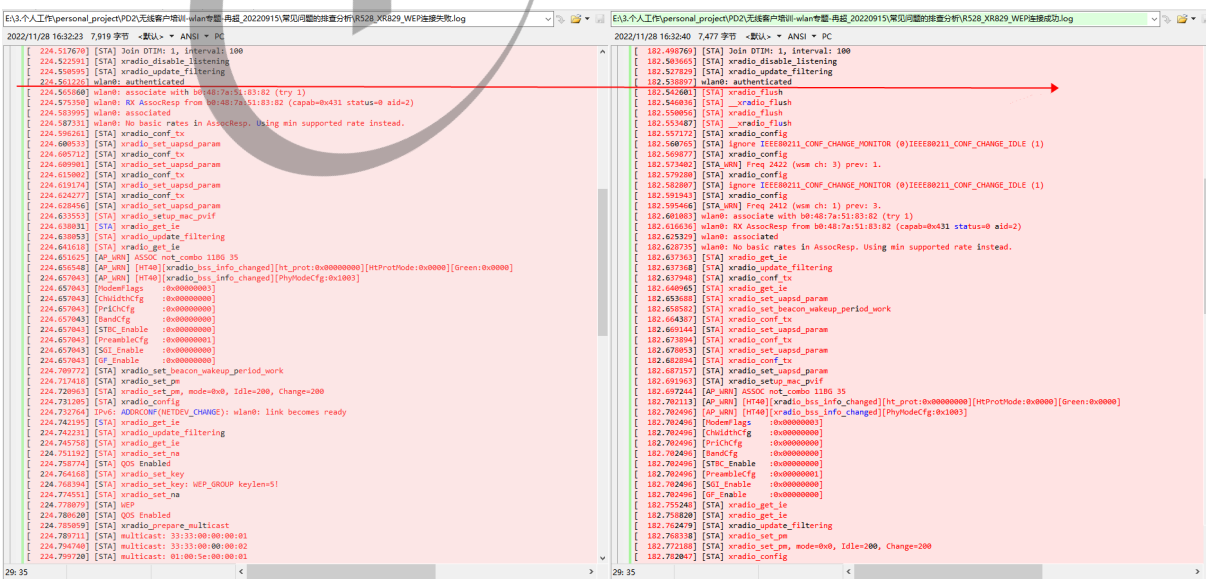

图 4-7: 4.3.4-3 日志对比

从日志对比分析我们可以看到在认证之后流程走的不一样，失败的日志是直接就发起了关联操

作。这里还不是很清楚，但根据打印基本定位到函数了，所以接着对比 umac/wep.c。

一对比就找到差异了。这部分的代码是关于 ICV 的操作。

图 4-8: 4.3.4-4 代码分析

根本原因：

tx 流程多次分配 IV 导致对方无法正常解析。

rx 流程没有正确去除 ICV，导致在 umac 层丢包。

解决办法：

xradio_set_key 中将 IEEE80211_KEY_FLAG_ALLOC_IV 标志去除，以避免多次分配 IV。

mac80211_crypto_wep_decrypt 中修改 if 条件判断，确保 skb 能上传。

思路总结：

1. 先做扫描测试。

2. 再做 ap 加密属性测试。

3. 做路由器兼容性测试。

4. 重新定位 wep 加密解析失败的环节：wifimanager,wpa_supplicant,driver。控制变量法

5. 根因分析驱动 wep 加密失败的点：对比分析法。

4.4 网络访问

4.4.1 排查思路

(1) 软件速查

1）使用wifi_scan_results_test/wifi -s/wpa_cli进行连接测试。详细命令参考《Tina_Linux_Wi-Fi_软件开发指南》。
注：如果不能正常连接网络，请参考4.3章节连接相关问题排查。
2）执行：ifconfig wlan0查看ip地址。
注：如果获取不了ip地址，检查配置：netifd，udhcpc配置，并尝试使用udhcpc -i wlan0手动获取ipi地址。
3）执行：ping路由器网关测试。
注：如果ping不同路由网关，检查网关地址是否同其他网口一样，导致无法路由出去。并参考4.3章节分析连接流程，确保真的已经连上路由。
4）执行：ping 局域网设备ip测试。
注：如果无法ping通局域网，检查局域网设备防火墙设置以及板子的路由表（使用route -n查看）。
5）执行：ping 外网ip地址测试。
注：如果无法ping通外网ip地址，检查板子路由表（使用route -n查看）。
6）执行：ping www.baidu.com域名测试。
注：如果无法ping通外网域名，检查路由器的外网访问能力（用手机进行测试），检查dns服务（查看/etc/resolv.conf），检查路由表(route -n查看)。
7）执行：route -n分析路由表。
8）分析/etc/resolv.conf。
9）打开wifimanager,wpa_supplicant,驱动（连接过程）的打印，连接测试抓日志分析。参考[第2章节]。
10）连接过程，Ping www.baidu.com抓包分析，参考《Tina_Linux_Wi-Fi_抓包使用指南》。

4.4.2 案例一：DNS 服务未开启导致无法 ping 通百度

问题描述：系统起来，利用 wifimanager 联网后，ping 百度不通。

问题背景：

产品：语音模块

主控：R329

模组：XR819

问题表现：系统联网后 ping 百度测试提示：ping: bad address ‘www.baidu.com’

问题分析：

1. 连通性测试。

- 测试 ping 路由正常。
- 测试 ping 百度的 ip 地址正常。
- 测试 ping 百度的域名异常。
- 测试其他设备 ping 百度域名正常。

基本可以确定就是 DNS 解析问题。

2. 抓包分析：

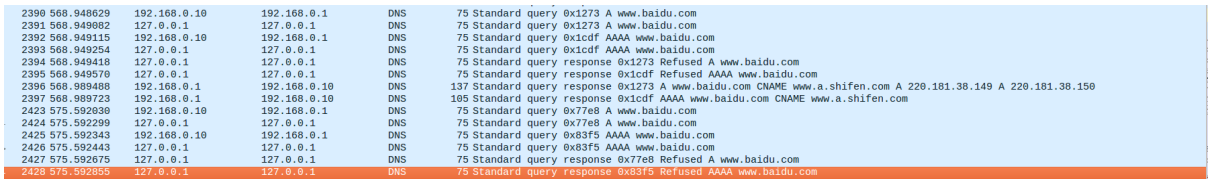

图 4-9: 4.4.2-1dns 抓包分析.png

从上面的抓包文件分析可以明显看到请求解析被拒绝了。dns 一版的查询顺序是：

主机/etc/host–>/etc/resolv.conf—>resolv.dnsmasq.conf，查看发现 host 和 resolv.conf 都为空，resolv.dnsmasq.conf 文件不存在，

分析配置文件：/etc/dnsmasq.conf

```
resolv-file=/etc/resolv.conf
strict-order
cache-size=102400
interface=wlan0
min-cache-ttl=3600
dhcp-range=192.168.5.2,192.168.5.255
all-servers
dhcp-sequential-ip
```

发现 dns 解析时的确是使用/etc/resolv.conf。于是手动添加网关作为 dns 服务器：nameserver 192.168.43.1。添加后测试正常。

根本原因：DNS 服务未开启，导致 ping 域名时 DNS 解析失败。

解决办法：

打开 DNS 配置：

```
make menuconfig
> Base system
<*> dnsmasq..... DNS and DHCP server
```

默认会开机自启动，(手动执行，一般是运行./etc/init.d/dnsmasq start)

编辑/etc/dnsmasq.conf，添加监听地址：

```
listen-address=127.0.0.1 # 如果用此计算机作为一组主机的默认 DNS，就需要使用固定IP地址：
listen-address=192.168.8.1 # 其它主机的dns设置使用这个ip为dns服务器(/etc/resolv.conf)。
listen-address=192.168.8.132,127.0.0.1 # 服务监听的网络接口地址。
```

思路总结：

1. 直接 ping 路由网关

- 直接 ping 路由器网关，确保设备正常连上了路由。

2. 直接 ping 外网的 ip 地址

- 直接 ping 外网的 ip 地址，分析是否是 DNS 失败导致无法 ping 外网域名，还是路由网关和设备 ip 一样导致的。

3.ping 外网域名

- ping 外网域名时，抓包分析。

4. 环境排查

- 找另外的设备对比测试，排除路由器问题。
- 确定是路由器问题的话，检查是否有网段限制。
- 检查是否存在路由网关和板卡的 wlan 接口 ip 地址一样。

信息反馈：

按照排查指南仍无法解决，给出当前初步分析定位的点，提供如下信息：

- 初步分析确定是 dns 问题 [提供 dns 配置/etc/resolv.conf/etc/dnsmasq.conf] 还是路由器兼容性问题 [提供路由型号]。
- 提供软硬件环境，便于复测排查。

4.4.3 案例二：路由表未更新导致无法访问外网

问题描述：ifconfig wlan0 down 然后再 ifconfig wlan0 up 设备无法 ping 通百度。

问题背景：

产品：标案

硬件：R528 + XR829

软件：Tina

问题表现：网卡 down 后 up 出现设备无法回连。

问题分析：

1. 复现问题

1) 执行 wifi -c allwinner-wpa2 Aa123456, 给设备联网。

```
root@TinaLinux:/# wifi -c allwinner-wpa2 Aa123456
2022-11-28 17:16:48:948: WMG_INFO [wifi_daemon.c:cmd_handle_c:646]:  ===Wi-Fi connect use sec(3)===
2022-11-28 17:16:48:948: WMG_INFO [wifi_daemon.c:cmd_handle_c:647]:  ===Wi-Fi connect successful,time
        7250.000000 ms===
root@TinaLinux:/# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr FC:02:C3:81:88:24
        inet addr:192.168.31.73  Bcast:192.168.31.255  Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:5030 errors:0 dropped:371 overruns:0 frame:0
        TX packets:1080 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:1763998 (1.6 MiB)  TX bytes:129608 (126.5 KiB)
```

2）执行 ping www.baidu.com, 确保设备联网成功并可以访问外网。

```
root@TinaLinux:/# ping www.baidu.com -c 3
PING www.baidu.com (163.177.151.110): 56 data bytes
64 bytes from 163.177.151.110: seq=0 ttl=51 time=28.860 ms
64 bytes from 163.177.151.110: seq=1 ttl=51 time=32.587 ms
64 bytes from 163.177.151.110: seq=2 ttl=51 time=44.966 ms
--- www.baidu.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 28.860/35.471/44.966 ms
```

3）执行 ifconfig wlan0 down, 关闭设备网卡。

```
root@TinaLinux:/# ifconfig wlan0 down
[ 6439.902333] wlan0: deauthenticating from 50:d2:f5:f1:b7:08 by local choice (reason=3)
wlan0: CTRL-EVENT-DISCONNECTED bssid=50:d2:f5:f1:b7:08 reason=3 locally_generated=1
[ 6439.965132] [WSM_WRN] Issue unjoin command(TX).
[ 6439.973212] [WSM_WRN] STA mode, send_deauth_to_self
[ 6439.978764] [TXRX_WRN] Issue unjoin command(TX) by self.
```

4）执行 ifconfig wlan0 up，再次打开设备网卡。

```
root@TinaLinux:/# ifconfig wlan0 up
[ 6569.746406] ieee80211_do_open: vif_type=2, p2p=0, ch=3, addr=fc:02:c3:81:88:24
[ 6569.754746] [STA] !!!xradio_vif_setup: id=0, type=2, p2p=0, addr=fc:02:c3:81:88:24
[ 6569.764567] [AP_WRN] BSS_CHANGED_ASSOC but driver is unjoined.
wlan0: SME: Trying to authenticate with 50:d2:f5:f1:b7:08 (SSID='allwinner-wpa2' freq=2437 MHz)[ 6571.324032] wlan0:
        authenticate with 50:d2:f5:f1:b7:08 (try 1)
[ 6571.479821] wlan0: authenticated
wlan0: Trying to associate with 50:d2:f5:f1:b7:08 (SSID='allwinner-wpa2' freq=2437 MHz)[ 6571.483935] wlan0: associate
        with 50:d2:f5:f1:b7:08 (try 1)
[ 6571.539219] wlan0: RX AssocResp from 50:d2:f5:f1:b7:08 (capab=0xc11 status=0 aid=2)
[ 6571.547872] wlan0: associated
wlan0: Associated with 50:d2:f5:f1:b7:08
wlan0: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
wlan0: WPA: Key negotiation completed with 50:d2:f5:f1:b7:08 [PTK=CCMP GTK=TKIP]
wlan0: CTRL-EVENT-CONNECTED - Connection to 50:d2:f5:f1:b7:08 completed [id=4 id_str=]
```

再次 up wlan0 时，设备认证，关联和四次密钥握手都完成了。

说明

为了方便调试，测试时把 wpa_supplicant 和驱动的日志都打开。

2. 网络连通性判断。

1）可以 ping 通路由器网关。

```
root@TinaLinux:/# ping 192.168.31.1 -c 3
PING 192.168.31.1 (192.168.31.1): 56 data bytes
64 bytes from 192.168.31.1: seq=0 ttl=64 time=49.531 ms
64 bytes from 192.168.31.1: seq=1 ttl=64 time=145.106 ms
64 bytes from 192.168.31.1: seq=2 ttl=64 time=24.277 ms

--- 192.168.31.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 24.277/72.971/145.106 ms
```

2）无法 ping 通百度。

```
root@TinaLinux:/# ping www.baidu.com
PING www.baidu.com (163.177.151.109): 56 data bytes
ping: sendto: Network unreachable
root@TinaLinux:/# ping 163.177.151.109
PING 163.177.151.109 (163.177.151.109): 56 data bytes
ping: sendto: Network unreachable
```

无法 ping 通外网，直接 ping 百度的 ip 地址【说明与 DNS 肯定无关】表现也一样无法 ping 通。

3）进一步打开驱动的调试和抓包发现都没有发 icmp 的包。

```
echo 0x2080 0x2080 > /sys/kernel/debug/ieee80211/phy0/xradio/parse_flags
```

4）怀疑是不是 ip 地址没有更新，于是手动实行 udhcpc -i wlan0 后测试，就可以 ping 通百度了。

```
root@TinaLinux:/# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr FC:02:C3:81:88:24
          inet addr:192.168.31.73  Bcast:192.168.31.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11027 errors:0 dropped:704 overruns:0 frame:0
          TX packets:1820 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4189738 (3.9 MiB)  TX bytes:214436 (209.4 KiB)
root@TinaLinux:/# udhcpc -i wlan0
udhcpc: started, v1.31.1
udhcpc: sending discover
udhcpc: sending select for 192.168.31.73
udhcpc: lease of 192.168.31.73 obtained, lease time 43200
udhcpc: ifconfig wlan0 192.168.31.73 netmask 255.255.255.0 broadcast 192.168.31.255
udhcpc: setting default routers: 192.168.31.1
```

虽然可以 ping 通百度了，但是仔细看经过 udhcpc 后 ip 地址并没有更新，还是之前的。看来 ip 地址不是问题的根因。[其实这里就应该去研究 udhcpc 到底做了啥]

5）使用局域网内的手机测试。

局域网内的手机可以 ping 通板子。板子无法 ping 通手机，但通过驱动打印和抓包可以看到有发出 icmp ping 包.

而且长时间 ping 测试发现实际上是可以间歇性 ping 通的，基本是约 1 分钟。

```
root@TinaLinux:/# ping 192.168.31.94
PING 192.168.31.94 (192.168.31.94): 56 data bytes
64 bytes from 192.168.31.94: seq=57 ttl=64 time=152.583 ms
[12469.501655] [XRADIO] if0-TX---ICMP(ping), Seq=57-192.168.31.73(s)-192.168.31.94(d)
[12469.645301] [XRADIO] if0-RX---ICMP(reply), Seq=57-192.168.31.94(s)-192.168.31.73(d)
...
64 bytes from 192.168.31.94: seq=116 ttl=64 time=83.448 ms
[12529.028973] [XRADIO] if0-TX---ICMP(ping), Seq=116-192.168.31.73(s)-192.168.31.94(d)
[12529.103361] [XRADIO] if0-RX---ICMP(reply), Seq=116-192.168.31.94(s)-192.168.31.73(d)
...
```

3. 怀疑路由表更新的问题，查看路由表。

1）查看一开始联网后的路由信息。

```
root@TinaLinux:/# wifi_connect_ap_test allwinner-wpa2 Aa123456
==================================
Connecting to the network(allwinner-wpa2).....
Connected to the AP(allwinner-wpa2)
/etc/rc.common: line 105: service_data: not found
Getting ip address(allwinner-wpa2).....
Wifi connect ap : Success!
==================================
root@TinaLinux:/# route -n
Kernel IP routing table
Destination    Gateway      Genmask      Flags Metric Ref    Use Iface
0.0.0.0        192.168.31.1   0.0.0.0      UG   0      0        0 wlan0
192.168.31.0   0.0.0.0      255.255.255.0  U    0      0        0 wlan0
root@TinaLinux:/# netstat -r
Kernel IP routing table
Destination    Gateway      Genmask      Flags   MSS Window  irtt Iface
default        XiaoQiang    0.0.0.0      UG        0 0          0 wlan0
192.168.31.0   *            255.255.255.0  U       0 0          0 wlan0
```

2）查看网卡关闭后再打开的路由信息。

```
root@TinaLinux:/# ifconfig wlan0 down
root@TinaLinux:/# route -n
Kernel IP routing table
Destination    Gateway      Genmask      Flags Metric Ref    Use Iface
root@TinaLinux:/# ifconfig wlan0 up
root@TinaLinux:/# route -n
Kernel IP routing table
Destination    Gateway      Genmask      Flags Metric Ref    Use Iface
192.168.31.0   0.0.0.0      255.255.255.0  U    0      0        0 wlan0
root@TinaLinux:/# netstat -r
```

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.31.0    *               255.255.255.0   U         0 0          0 wlan0
```

从这里确实可以看到经历过一次 wlan0 down 后再 wlan0 up，外网的网关没有了。

```
route -n各字段说明：
destination：目的网段，0.0.0.0 表示缺省路由表，优先级最高；
mask：与网络目标地址相关联的网掩码（又称之为子网掩码）。主机路由 255.255.255.255 ，默认路由是 0.0.0.0。
Iface：到达该目的地的本机 interface
gateway：网关，下一跳路由器入口的 ip，路由器通过 interface 和 gateway 定义到下一个路由器的链路。
metric 跳数(路由质量)：一般情况下，如果有多条到达相同目的地的路由记录，优先采用metric值小的那条路由。
```

3）手动设置一次外网网关

```
root@TinaLinux:/# ping www.baidu.com
PING www.baidu.com (163.177.151.109): 56 data bytes
ping: sendto: Network unreachable
root@TinaLinux:/# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.31.0    0.0.0.0         255.255.255.0   U     0      0        0 wlan0
root@TinaLinux:/# route add default gw 192.168.31.1
root@TinaLinux:/# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.31.1    0.0.0.0         UG    0      0        0 wlan0
192.168.31.0    0.0.0.0         255.255.255.0   U     0      0        0 wlan0
root@TinaLinux:/# ping www.baidu.com -c 3
PING www.baidu.com (163.177.151.110): 56 data bytes
64 bytes from 163.177.151.110: seq=0 ttl=51 time=32.434 ms
64 bytes from 163.177.151.110: seq=1 ttl=51 time=35.709 ms
64 bytes from 163.177.151.110: seq=2 ttl=51 time=44.778 ms
--- www.baidu.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 32.434/37.640/44.778 ms
```

可以 ping 通百度了。回过头来再去分析 udhcpc 的动作。

4）研究 udhcpc 的动作

```
 1 #!/bin/sh
 2 [ -z "$1" ] && echo "Error: should be run by udhcpc" && exit 1
 3
 4 RESOLV_CONF="/etc/resolv.conf"
 5
 6 set_classless_routes() {
 7   local max=128
 8   local type
 9   while [ -n "$1" -a -n "$2" -a $max -gt 0 ]; do
10     [ ${1##*/} -eq 32 ] && type=host || type=net
11     echo "udhcpc: adding route for $type $1 via $2"
12     route add -$type "$1" gw "$2" dev "$interface"
13     max=$(($max-1))
14     shift 2
15   done
16 }
```

```
17
18 setup_interface() {
19   echo "udhcpc: ifconfig $interface $ip netmask ${subnet:-255.255.255.0} broadcast ${broadcast:-+}"
20   ifconfig $interface $ip netmask ${subnet:-255.255.255.0} broadcast ${broadcast:-+}
21
22   [ -n "$router" ] && [ "$router" != "0.0.0.0" ] && [ "$router" != "255.255.255.255" ] && {
23     echo "udhcpc: setting default routers: $router"
24
25     local valid_gw=""
26     for i in $router ; do
27       route add default gw $i dev $interface
28       valid_gw="${valid_gw:+$valid_gw|}$i"
29     done
30
31     eval $(route -n | awk '
32       /^0.0.0.0\W{9}('$valid_gw')\W/ {next}
33       /^0.0.0.0/ {print "route del -net "$1" gw "$2";"}
34     ')
35   }
37   set dns server
38   echo -n > $RESOLV_CONF
39   for i in $dns ; do
40     echo nameserver $i >> $RESOLV_CONF
41   done
42
43   # CIDR STATIC ROUTES (rfc3442)
44   [ -n "$staticroutes" ] && set_classless_routes $staticroutes
45   [ -n "$msstaticroutes" ] && set_classless_routes $msstaticroutes
46 }
49 applied=
50 case "$1" in
51   deconfig)
52     ifconfig "$interface" 0.0.0.0
53   ;;
54   renew)
55     setup_interface update
56   ;;
57   bound)
58     setup_interface ifup
59   ;;
60 esac
61
62 # user rules
63 [ -f /etc/udhcpc.user ] && . /etc/udhcpc.user
64
65 exit 0
```

这里就破案了，原来 udhcpc 不只是做了更新 ip 地址，还设置了网关，设置了 DNS。

根本原因：

Ifconfig down wlan0 关闭了网卡，ifconfig wlan0 up 重新打开网卡时没有设置外网网关。

解决办法：

1. 手动调用 udhcpc，借用脚本里面的设置网关的动作。

2. 手动执行：route add default gw 192.168.31.1，设置网关。

思路总结：

1. 复现问题。

2. 从网关，局域网，外网 ip，外网域名，逐步扩大范围检查连通性。

3. 驱动日志，抓包，对比分析。

4. 定位分析：无法 ping 外网 ip 地址，路由转发失败。

4.4.4 案例三：ifconfig 无法同步 ip 地址导致访问网络失败

问题描述：无法访问网络，局域网和外网都不行。

问题背景：

产品：标案

硬件：R528 + XR829

软件：Tina

问题表现：

1. 使用 wpa_cli 给设备联网。

2. 执行 ping www.baidu.com, 访问外网失败。

3. 执行 ping 192.168.31.1 ping 网关也失败。

问题分析：

1. 复现问题，按照如上步骤测试。

1.1 加载驱动，准备 wpa_supplicant.conf 文件。

```
insmod /lib/modules/5.4.61/xr829.ko
cp /etc/wifi/wpa_supplicant/wpa_supplicant* /tmp
wpa_supplicant -iwlan0 -Dnl80211 -c/tmp/wpa_supplicant.conf -d -I/tmp/wpa_supplicant_overlay.conf -O/tmp/sockets
        -B
```

1.2 加载服务

```
wpa_supplicant -iwlan0 -Dnl80211 -c/tmp/wpa_supplicant.conf -d -I/tmp/wpa_supplicant_overlay.conf -O/tmp/sockets
        -B
```

1.3 使用 wpa_cli 配网

```
wpa_cli -i wlan0 -p /tmp/sockets/ scan;
wpa_cli -i wlan0 -p /tmp/sockets/ scan_results;
wpa_cli -i wlan0 -p /tmp/sockets/ add_network;
wpa_cli -i wlan0 -p /tmp/sockets/ set_network 0  ssid '"allwinner-wpa2"';
```

```
wpa_cli -i wlan0 -p /tmp/sockets/ set_network 0  key_mgmt WPA-PSK;
wpa_cli -i wlan0 -p /tmp/sockets/ set_network 0  psk '"Aa123456"';
wpa_cli -i wlan0 -p /tmp/sockets/ save_config;
wpa_cli -i wlan0 -p /tmp/sockets/ select_network 0;
wpa_cli -i wlan0 -p /tmp/sockets/ enable_network 0;
wpa_cli -i wlan0 -p /tmp/sockets/ list_network;
wpa_cli -i wlan0 -p /tmp/sockets/ status;
```

1.4 获取 ip 地址

```
udhcpc -iwlan0
```

2. ping 测试

2.1 可以 ping 百度域名测试。

```
root@TinaLinux:/# ping www.baidu.com
ping: bad address 'www.baidu.com'
```

2.2 ping 百度 ip 地址测试。

```
root@TinaLinux:/# ping 163.177.151.109
PING 163.177.151.109 (163.177.151.109): 56 data bytes
ping: sendto: Network unreachable
```

2.3 ping 路由网关测试

```
root@TinaLinux:/# ping 192.168.31.1
PING 192.168.31.1 (192.168.31.1): 56 data bytes
ping: sendto: Network unreachable
```

都无法 ping 通，猜测是没有路由器根本就没有转发，分析路由表。

3. 查看路由表

```
root@TinaLinux:/# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
```

果然都没有路由表，再执行 ifconfig 查看

```
root@TinaLinux:/# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr A8:3A:D7:53:79:61
          inet6 addr: fe80::aa3a:d7ff:fe53:7961/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2584 errors:0 dropped:0 overruns:0 frame:0
          TX packets:23 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:109742 (107.1 KiB)  TX bytes:2914 (2.8 KiB)
```

发现 wlan0 居然没有 ip 地址，可是前面执行 udhcpc 时明明就成功获取到了 ip 地址了啊。

4. 打开驱动调试进一步测试，再次尝试获取 ip 地址。

```
echo 0x20c0 0x20c0 > /sys/kernel/debug/ieee80211/phy0/xradio/parse_flags
```

```
root@TinaLinux:/# echo 0x20c0 0x20c0 > /sys/kernel/debug/ieee80211/phy0/xradio/p
arse_flags
[  963.211444] [XRADIO] txparse=0x20c0, rxparse=0x20c0
root@TinaLinux:/# udhcpc -i wlan0
udhcpc: started, v1.27.2
udhcpc: sending discover
[  967.799449] [XRADIO] if0-TX---DHCP, Opt=53, MsgType=1-0.0.0.0(s)-255.255.255.255(d)
[  967.821676] [XRADIO] if0-RX---DHCP, Opt=53, MsgType=2-192.168.31.1(s)-192.168.31.152(d)
udhcpc: sending select for 192.168.31.152
[  967.889422] [XRADIO] if0-TX---DHCP, Opt=53, MsgType=3-0.0.0.0(s)-255.255.255.255(d)
[  967.914309] [XRADIO] if0-RX---DHCP, Opt=53, MsgType=5-192.168.31.1(s)-192.168.31.152(d)
udhcpc: lease of 192.168.31.152 obtained, lease time 43200
```

到这基本就可以确定问题的根因了，ip 地址没有同步到系统 wlan0，自然就无法进行 Ping 测试了。

5. 尝试手动设置

```
ifconfig wlan0 192.168.31.152 broadcast 192.168.31.255 netmask 255.255.255.0
route add default gw 192.168.31.1
```

再进行测试发现可以 ping 通网关，ping 通局域网内的设备，也能 Ping 通百度的 ip 地址，但是不能 Ping 通百度域名。由此看来 dns 还有问题，也需要手动设置一下。

到这里基本可以断定问题所在了，在 tina 系统上，同步 ip 地址和 dns 服务都是依靠/usr/share/udhcpc/default.script 做的，猜测：

5.1 没有配置 netifd 和 udhcpc 导致.

5.2 文件系统只读。

```
> Base system
 -*- netifd..... OpenWrt Network Interface Configuration Daemon
> Base system > busybox
 [*] udhcpc (DHCP client) (NEW)
```

6. 验证实验

尝试去/etc/目录下创建文件，果然失败。系统果然没有配置 netifd。

根本原因：

- 客户没有使用 wifimanager, 将依赖配置项 netifd 和 udhcpc 忘记配置了。
- 当前系统是只读的。

解决办法：

1. 系统 tina 配置 netifd。

2. 将 wpa_supplicant.conf 以及 resolv.conf 移动到/tmp 目录或者修改文件系统可读写。

思路总结：

1. 复现问题。

2. 从网关，局域网，外网 ip，外网域名，逐步扩大范围检查连通性。

3. 驱动日志，抓包，对比分析。

4. 定位分析：重点分析 ip 地址，路由表，dns 服务。

4.4.5 案例四：ipv6 dns 解析耗时导致 ping 百度延时高

问题描述：设备联网后 ping 百度延时高问题

问题背景：

产品：智能音箱

硬件：R329+RTL8733BS

软件：Tina4.0+Wi-Fi

复现概率：必现，ping 百度时抓包分析可以看到 DNS 的回复约耗时 5s。

驱动版本：v5.14.1-47-6-g18fd19d93.20220301_COEX20211210-2706_zeroconfig_sc_0408

问题表现：

1. 复现步骤

1）设备系统默认起来会加载驱动。

2）给设备配网。

3）配网后执行 ping www.baidu.com

4）利用 Omnipeek 进行抓包。

2. 具体表现

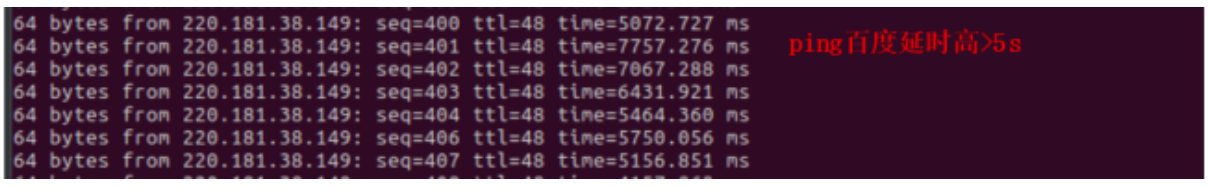

图 4-10: 4.4.5-1ping 百度延时高

问题分析：

1. 问题复现

实验一：烧录最新固件，系统上电，加载驱动和 wpa_supplicant 服务，并 up wlan0，给设备配网后，执行 ping www.baidu.com 进行压测。

2. 问题定位

实验二：利用 Tina 自带 wifimanager 应用进行联网后测试。

问题仍然复现，可以排除应用的影响。

实验三：同环境下使用其他 XR829 的板子测试。

问题仍然出现，可以排除 wifi 模组的影响。

实验四：使用该设备 ping 网关测试，ping 其他域名，www.sina.com.cn, www.youku.com 测试

Ping 网关时延时 <50ms，ping 新浪和优酷的域名同百度一样，延时高于 5s。

经过上面的分析可以定位是域名解析引入了延时的差异。

实验五：ping www.baidu.com 抓包分析解析过程

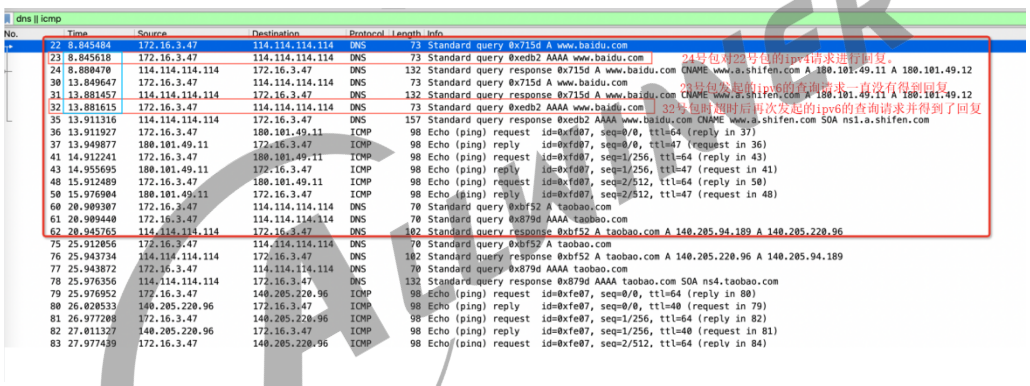

图 4-11: 4.4.5-2ping 百度抓包

从分析可以发现，主要耗时在 AAAA 的请求回复（即 IPV6 的解析），同环境下对正常的设备抓包分析如下：正常设备 ping 百度抓包分析：

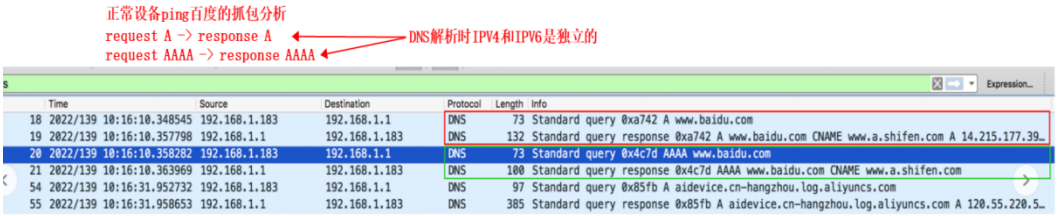

图 4-12: 4.4.5-3ping 百度正常抓包

异常设备 ping 百度抓包分析：

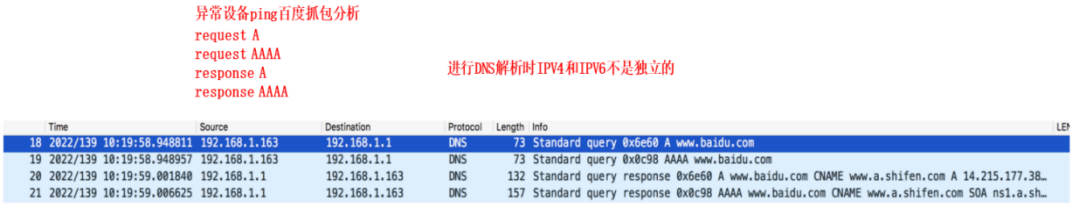

图 4-13: 4.4.5-4ping 百度异常抓包

两个包文件对比分析看到差异了，异常设备 DNS 解析是同时对 IPV4，IPV6 进行的，而且不是独立的；

而正常设备 ping 百度延时低，DNS 解析也是同时对 IPV4，IPV6 进行，但两者是相互独立的。

从以上分析可以得出如下解决办法：

方案一：根据设备对 IPV6 的实际需求，取消 IPV6 的 DNS 解析。

方案二：DNS 解析时，IPV4，IPV6 独立解析。

根本原因：

当前系统是同时发起 IPV4，IPV6 的 DNS 解析，并发处理时必须等 IPV6 解析完成 DNS 的解析结果才会返回，导致 ping 百度延时高。

解决办法：

考虑到设备日后的兼容性，这里采用方案二：DNS 解析时，IPV4，IPV6 独立进行。

可以在 DNS 的配置文件【/etc/resolv.conf】中进行修改：

```
options single-request-reopen timeout:1 attempts:1
single-request-reopen：DNS解析时IPv4和IPv6采用独立请求，互不影响。如果做IPv6解析一般会失败，耗时很久。
timeout：DNS解析失败的超时时长
attempts：失败重试的次数
```

Tina 系统在/etc/resolv.conf 中添加 [通过 package/network/config/netifd/files/usr/share/udhcpc/default.script]

```
diff --git a/network/config/netifd/files/usr/share/udhcpc/default.script b/network/config/netifd/files/usr/share/udhcpc/
        default.script
index 8c5dce13..34528b36 100755
--- a/network/config/netifd/files/usr/share/udhcpc/default.script
+++ b/network/config/netifd/files/usr/share/udhcpc/default.script
@@ -37,9 +37,11 @@ setup_interface() {

        set dns server
        echo -n > $RESOLV_CONF
+       echo "options single-request-reopen" > $RESOLV_CONF
        for i in $dns ; do
          echo nameserver $i >> $RESOLV_CONF
        done
+       echo "options attempts:1 timeout:5" >> $RESOLV_CONF
```

```
# CIDR STATIC ROUTES (rfc3442)
[ -n "$staticroutes" ] && set_classless_routes $staticroutes
```

思路总结：

现场确认。

- 确认网络当前的状态，是否仍然保持连接，且可以 ping 通百度。
- ping 域名测试。
- ping IP 地址测试。
- 分析 dns。

信息反馈：

按照排查指南仍无法解决，给出当前初步分析定位的点，提供如下信息：

- 若网络 ping 百度域名的提供日志，以及抓包文件。

4.4.6 案例五：在线视频播放掉线

问题描述：在线播放音乐压测异常掉线。

问题背景：

产品：公板

主控：R329

模组：XR829

问题表现：利用 wifimanager 联网后，tplayer 播放在线视频，异常掉线。

问题分析：

现场确认，此时 ping 百度还正常，说明设备仍然和路由保持连接。

kill 掉 tplayer 线程后重新播放恢复正常。

分析测试日志没有发现异常打印。

怀疑点：

- 网络中途断网，之后又恢复了。
- 局域网内存在 mac 地址冲突，设备测试过程中被其他设备挤掉线。

根本原因：局域网内多设备存在 mac 地址冲突。

解决办法：

统一给测试板卡烧录 mac 地址。

使用随机 mac 地址。

思路总结：

现场确认。

- 确认网络当前的状态，是否仍然保持连接，且可以 ping 通百度。
- 如果已经断网，确认是否可以再次成功连接网络。
- 用其他设备连接该路由，确认外网正常。
- 确认周边环境是否存在 mac 地址冲突的设备。

日志分析。

- 分析日志是否有异常，特别是驱动断网打印。

信息反馈:

按照排查指南仍无法解决，给出当前初步分析定位的点，提供如下信息：

- 若网络已经断开连接的，提供日志，以及抓包文件。
- 若网路保持连接的，确保周围设备 mac 地址不冲突，外网可正常访问。
- 提供软硬件环境，便于复测分析。

4.5 休眠唤醒

4.5.1 排查思路

一、软件速查

1）使用 wifi_scan_results_test/wifi -s/wpa_cli 进行连接测试。详细命令参考《Tina_Linux_Wi-Fi_ 软件开发指南》。

执行：ifconfig 查看 ip 地址。

说明

注：如果不能正常连接网络，请参考 4.3 章节连接相关问题排查。

2）执行：echo mem > /sys/power/state 进入休眠。

说明

注：此时串口不能在输入输出。

3）基于 2 正常休眠的前台下，尝试用其他唤醒源唤醒，比如按键，确保休眠唤醒基本功能正常。

4）基于 3/4 的前提下尝试局域网 ping 设备的 ip 地址唤醒。

二、软件细查

休眠唤醒问题划分为休眠和唤醒两个过程，基于如上测试可以缩小范围。

5）若是无法休眠

```
echo 0 > /sys/kernel/autohotplug/enable;
echo 1 >  /sys/power/pm_print_times;
echo N > /sys/module/printk/parameters/console_suspend;
echo Y > /sys/module/kernel/parameters/initcall_debug;
echo 8 > /proc/sys/kernel/printk;
```

如上打开调试，并参考第 2 章打开驱动的打印后抓取日志分析。

分析关键字：suspend 时被那个设备阻断，我们重点关系 wlan，所以可以做一个对比实验：卸载掉 wlan 驱动后休眠。

如果能够正常休眠，说明 wlan 驱动相关，目前没有通用的解题思路，将抓取的日志反馈。

如果同样不能正常休眠，说明与 wlan 驱动无关。

6）若是可以正常休眠，但是无法唤醒。

基于 3，可以确定休眠唤醒框架正常。

```
6.1）通过路由器后台观察设备此时是否还正常保持连接。
6.2）电压表测量休眠后wlan设备的供电情况。
6.3）示波器测量wlan设备的clk波形。
```

4.5.2 案例一：ping 唤醒失败

问题描述：联网后系统休眠，无法通过 ping 的方式唤醒主控。

问题背景：

产品：公板

主控：R528

模组：XR829

问题表现：系统联网后进入休眠，局域网内 ping 板子的 ip 地址，系统无法唤醒。

问题分析：

1. 使用 gpio PE2 做个按键可以唤醒，rtc 的方式设置定时唤醒也可以正常唤醒。说明系统休眠唤醒框架正常。

2. 通过路由器查看休眠后板子已经断开连接了。说明已经不是保活模式了。
3. 进一步也分析了唤醒引脚 PE6 的配置，中断模式，高电平触发，且中断号和 PE5 是同一组，尝试强制拉高 PE6 可以唤醒。
4. 该方案采用的是 fanout 的功能来输出 32k 波形，休眠前可以测量到 32k，休眠后无法测量到。基本定位就是这里的问题了。

根本原因：

当前 wifi 的 32k 来源 PG11 的 fanout 功能，而 clk_fanout 功能依赖 pll_peri , 但使能 pll_peri 又依赖于 pll_ldo，系统休眠时会直接关闭 pll_ldo，所有导致无法产生 32k，最后 wifi 保活失败，所有无法 ping 唤醒。

解决办法：

1. 休眠时不关闭 pll_ldo。–》会导致功耗增加。
2. 切换时钟：rtc32k->apb0->pclk->fanout2 的通路为 WIFI 模块提供 32K 时钟。

思路总结：

1. 其他唤醒源测试

- 休眠唤醒涉及应用，设备，cpu 核等多个阶段的操作，wifi 的休眠唤醒配合系统休眠框架处于设备一层。所有必须确保休眠唤醒框架本身正常，才好近一步分析。一般的思路就是先用其他唤醒源测试，例如 gpio, 按键，rtc 等测试是否可以唤醒。确保系统休眠唤醒框架正常。

2. 确定休眠时 wifi 状态

- wifi ping 唤醒，显然此时网络是要保持连接的，否则局域网肯定没法 ping 通。可以通过连接的路由器后台查看板卡状态，甚至通过抓包查看路由和板子直接是否有正常的 probe 和 null data 帧。

3. 确定唤醒引脚状态

- ping 唤醒的本质也是 gpio 中断，可以产看当前唤醒引脚的配置，是否是中断模式，中断是否使能，中断触发方式是否设置，所在 gpio 的中断号是否打开。必要时使用示波器抓取波形查看是否产生了中断。

4. 其他排查

- 时钟排查，休眠时 wifi 的 32k 时钟是一定要的。

信息反馈: 按照排查指南仍无法解决，给出当前初步分析定位的点，提供如下信息：

- 休眠后，wifi 是否保活，ping 是否触发了中断；直接通过 wifi 唤醒引脚强制触发中断是否可以唤醒。
- 提供软硬件环境，便于分析排查。

4.6 吞吐性能

4.6.1 排查思路

一、软件速查

1）吞吐测试方法参考《Tina_Linux_Wi-Fi_ 测试报告》。

说明

注：一定要确保板子和测试的笔记本可以相互 Ping 通。

二、硬件排查

2）优先确保硬件 RF 指标正常，查看板卡的 RF 测试报告。

3）确保使用的板卡硬件正常，比如：板载天线是否有接通路电阻，外置天线是否有接，sdio 上下拉电阻是否 OK。

原理图是否有经过审核，尤其是涉及电平转换的电路。

三、软件细查

4）优先测试干净环境下的性能，比如屏蔽房环节。

4）用控制变量法分析，排除单一板卡差异，路由兼容性问题，网络访问控制差异，环境干扰影响等。

5）用对比分析法，寻找竞品或者前后版本做对比分析。

6）用抓大头方法进行优化，优先优化那些耗时多，占用多，影响大的环节。

说明

注：重点关注：
CPU 核数，CPU 频率，CPU 占用率【使用 cpu_monitor 工具】
线程优先级，SDIO 频率【参考第 2 章节】
测试 AP 的带宽设置/频段设置【路由后台设置查看】

4.6.2 案例一：吞吐测试无法建立连接

问题描述：吞吐测试时，板子 TX 无法建立连接。

问题背景：

产品：公板

主控：R328

模组：XR829

问题表现：

吞吐测试时无法建立连接，提示如下：

```
root@TinaLinux:/# iperf -c 192.168.31.100 -i 1 -t 20
connect failed: No route to host
```

问题分析：

测试板子和 pc 相互 ping，pc 可以 ping 通板子，板子无法 ping 通 pc。

替换另外的板卡测试表现一致。

替换 pc 测试正常。

根本原因：测试的 pc 开了防火墙。

解决办法：关闭 pc 的防火墙。直接用两块板卡进行测试。

思路总结：

1. 设备双方相互 ping 测试。

- 在做吞吐测试前，先确保设备双方可以相互 ping 通。

2.TX/RX 交换做吞吐测试。

- TX/RX 交换测试，确定是哪一方无法建立连接，还是双方都无法建立连接。

3. 环境排查。

- 设别是否存在 mac 地址和 ip 地址冲突，用于测试的笔记本是否开了防火墙。
- 对比测试：替换 pc/路由和板卡进行对比测试。

信息反馈:

按照排查指南仍无法解决，给出当前初步分析定位的点，提供如下信息：

- 排除环境因素：确保不存在 mac 地址，ip 地址的冲突，不存在防火墙等因素，定位是否是单一板卡，路由问题。
- 确定是单一模组，单一路由问题，提供软硬件环境，便于分析。

4.6.3 案例二：HT40 测试吞吐 TCP 不达标

问题描述：吞吐测试 TCP HT40 不达标。

问题背景:

产品：扫描笔

硬件：R853+XR829+ 个人 pc + 小米 A4 路由

软件：Tina4.0+Wi-Fi+iperf2

问题表现:

1. 复现步骤

1）使用 tina_r528s2-evb3hmi_uart0.img 方案编译固件烧录，系统默认起来会加载驱动。

2）路由器设置 HT40

3）使用个人 pc 和 iperf2 进行吞吐测试。

2. 具体表现

吞吐数据不达标，只有 50-60Mbps。

问题分析:

测试一：利用 R853+XR829 在屏蔽房进行吞吐测试：

序号	场景描述	路由设置	TCP-TX（Mbps）	TCP-RX(Mbps)	Cpumonitor
1	单核 1.1G	HT40	54/60/65	44/55/61.2	1104-100%
2	单核 1.2G	Ch 6	55/62/68.2	55/59/62.9	1200-96%

图 4-14: 4.6.3-1 吞吐数据

测试结论：

1）R853+XR829 测试 HT40 吞吐时，不达标 (低于 70Mbps)。

2）R853+XR829 测试 HT40 吞吐时，提升 CPU 频率从 1G 到 1.2G 有些许改善。

3）R853+XR829 测试 HT40 吞吐时，CPU 占用量很高，几乎是满带宽。

4）R853+XR829 测试 HT40 吞吐时，主要占用 CPU 的有：iperf，xradio_bh，ksoftirqd，xradio_proc。

4.2 问题定位

测试二：利用 R853+RTL8723DS 在屏蔽房进行吞吐测试：

序号	场景描述	路由设置	TCP-TX（Mbps）	TCP-RX(Mbps)	Cpumonitor
3	单核 1.1G	HT40	74.4/76.5/79.7	76.7/80.8/81.6	1104-85%
4	单核 1.2G	Ch 6	78.6/82.8/87	79.1/80/80.5	1200-85%

图 4-15: 4.6.3-2 吞吐数据

测试结论：

1）R853+RTL8723DS 测试 HT40 吞吐时，达标 (TCP TX/RX 都高于 70Mbps)。

2）R853+RTL8723DS 测试 HT40 吞吐时，提升 CPU 频率到 1.2G 有些许改善，但不明显，还是达标。

3）R853+RTL8723DS 测试 HT40 吞吐时，CPU 占用量也很高，但比 XR829 低。

4）R853+RTL8723DS 测试 HT40 吞吐时，主要占用 CPU 的有：iperf，RTW_RECV_THREAD，RTWHALXT，RTW_XMIT_THREAD，ksdioirq/mmc2。

5）和测试 1 对比，问题偏向于无线模组。

测试三：利用 R528+XR829 在屏蔽房进行吞吐测试

序号	场景描述【XR829】	路由设置	TCP-TX（Mbps）	TCP-RX(Mbps)	Cpumonitor
5	双核 1.2G	HT40	72.4/79.7/84.9	73.1/75.4/76.5	1200-56%, 1200-25%
6	单核 1.2G	Ch 6	58.7/59.8/60.8	53.9/54.1/55.6	1200-86%

图 4-16: 4.6.3-3 吞吐数据

测试结论：

1）R528+XR829 测试 HT40 吞吐时，双核达标，单核不达标，说明与主控关联不大。

2）R528+XR829 测试 HT40 吞吐时，双核变单核，吞吐降低约 15-25Mbps, 降低约 18%-28%

3）吞吐受 cpu 核数影响很大。

测试四：利用 R528+RTL8723DS 在屏蔽房进行吞吐测试

序号	场景【RTL8723DS】	带宽	TCP-TX (Mbps)	TCP-RX(Mbps)	Cpumonitor
7	双核 1.2G	HT40	74.4/75.6/78.6	83.6/85.7/88.2	1200-59%, 1200-52%
8	单核 1.2G		70.3/72.4/74.4	74.1/76.4/78.3	1200-76%

图 4-17: 4.6.3-4 吞吐数据

测试结论：

1）R528+RTL8723DS 测试 HT40 吞吐时，双核达标，单核也达标，说明主要受模组影响。

2）R528+RTL8723DS 测试 HT40 吞吐时，双核变单核，吞吐降低约 5-10Mbps, 降低约 5%-11%。

3）该问题的吞吐主要影响面：无线模组、CPU 核数。

测试五：利用 R853+XR829 在屏蔽房进行吞吐测试

考虑到 CPU 占用高，查看当前部分线程的优先级：

1. 查看 RTL8723DS 的 RTWHALXT, RTW_XMIT_THREAD,RTW_RECV_THREAD 的线程优先级

```
rtl8723ds/core/rtw_xmit.c
rtw_xmit_thread()
{
   struct sched_param param = { .sched_priority = 1 };
   sched_setscheduler(current, SCHED_FIFO, &param);
}
rtl8723ds/core/rtw_recv.c
rtw_recv_thread()
{
   struct sched_param param = { .sched_priority = 1 };
   sched_setscheduler(current, SCHED_FIFO, &param);
}
rtl8723ds/hal/rtl8723d/sdio/rtl8723ds_xmit.c
rtl8723ds_xmit_thread()
{
   struct sched_param param = { .sched_priority = 1 };
   sched_setscheduler(current, SCHED_FIFO, &param);
}
linux-5.4/include/uapi/linux/sched.h
 80 /*
 81  * Scheduling policies
 82  */
 83 #define SCHED_NORMAL      0
 84 #define SCHED_FIFO     1
 85 #define SCHED_RR       2
 86 #define SCHED_BATCH    3
 87 /* SCHED_ISO: reserved but not implemented yet */
 88 #define SCHED_IDLE     5
 89 #define SCHED_DEADLINE     6
```

查看 xr829 的 xradio_bh,xradio_proc 的线程优先级

```
xr829/wlan/bh.c
xradio_bh()
{
   ret = sched_setscheduler(hw_priv->bh_thread, SCHED_FIFO, &param);
}
```

```
xradio_proc()
{
   ret = sched_setscheduler(hw_priv->proc.proc_thread,SCHED_FIFO, &param);
}
linux-5.4/include/uapi/linux/sched.h
 80 /*
 81  * Scheduling policies
 82  */
 83 #define SCHED_NORMAL      0
 84 #define SCHED_FIFO     1
 85 #define SCHED_RR       2
 86 #define SCHED_BATCH    3
 87 /* SCHED_ISO: reserved but not implemented yet */
 88 #define SCHED_IDLE     5
 89 #define SCHED_DEADLINE     6
```

两个模组的线程优先级设置都一样，都是 SCHED_FIFO。

序号	场景	带宽	TCP-TX（Mbps）	TCP-RX(Mbps)	Cpumonitor
9	【R853+XR829】单核 1.2G， 线程优先级调整为 SCHED_RR（2）	HT40 CH6	56.6/65/68.2	55.6/62.4/65.3	1200-83%
10	【R853+XR829】单核 1.2G， 线程优先级调整为 SCHED_BATCH（3）		59.8/64/67.1	59.6/60.8/62.6	1200-94%

图 4-18: 4.6.3-5 吞吐数据

测试结论：

1）R853+XR829 测试 HT40 吞吐时，调整 xradio_bh 和 xradio_proc 的线程优先级到 SCHED_RR 后，有改善，但仍然不达标。

2）R853+XR829 测试 HT40 吞吐时，调整 xradio_bh 和 xradio_proc 的线程优先级到 SCHED_BATCH 后，有改善，但仍然不达标。

3）RTL8723DS/XR829 的线程优先级设置都一样，说明该问题线程优先级不是主要影响面，有一定提升。

测试六：利用 R853+XR829 在屏蔽房进行吞吐测试

调整 ampdu:

序号	场景	带宽	TCP-TX（Mbps）	TCP-RX(Mbps)	Cpumonitor
1	【R853+XR829】单核 1.2G 默认线程优先级 SCHED_FIFO（1） Ampdu 调整 8/10/12/16	HT40 CH6	49.3/59.8/65	42.5/60.7/64.5	1200-99%

图 4-19: 4.6.3-6 吞吐数据

测试结论：

1）修改 ampdu 后，吞吐没有提升，说明该问题 ampdu 不关联。

测试七：利用 R853+XR829 在屏蔽房进行吞吐测试

调整为 SDIO 中断:

序号	场景	带宽	TCP-TX（Mbps)	TCP-RX(Mbps)	Cpumonitor
1	【R853+XR829】单核 1.2G 默认线程优先级 SCHED_FIFO（1） 调整为 sdio 中断	HT40 CH6	56.6/66.1/68.2	52.3/58/60.5	1200-96%

图 4-20: 4.6.3-7 吞吐数据

测试结论：

1）使用 sdio 中断，吞吐没有提升，说明 sdio 中断和 gpio 中断没有差异。

测试八：利用 R853+XR829 在屏蔽房进行吞吐测试

调整 PG 口的 debounce, 选择 24M:

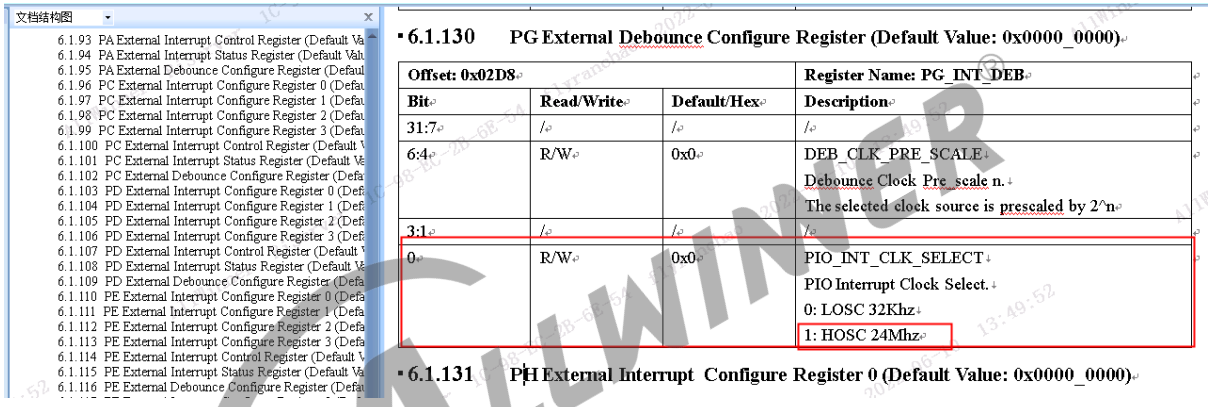

图 4-21: 4.6.3-8debounce

```
root@Tina:/sys/class/sunxi_dump# echo 0x020002d8 0x00000001 > write
root@Tina:/sys/class/sunxi_dump# cat dump
0x00000001
```

图 4-22: 4.6.3-8sunxi-dump

序号	场景	带宽	TCP-TX（Mbps)	TCP-RX(Mbps)	Cpumonitor
1	【R853+XR829】单核 1.2G 默认线程优先级 SCHED_FIFO（1） 调整 PG 口 debounce 采用 24M clk	HT40 CH6	53.5/55.6/58.7	59.3/63/64.2	1200-96%

图 4-23: 4.6.3-8 吞吐数据

测试结论：1）调整 io 的的 clk debounce，吞吐没有提升，说明与 io 口 clk debounce 无关联。

测试九：利用 R853+XR829 在屏蔽房进行吞吐测试确认是否有开 riscv 的核，已经确认默认没有开

```
[package]
;item=Item_TOC_name,         Item_filename,
;item=scp,                    scp.fex
item=optee,                  optee.fex
item=u-boot,                 u-boot.fex
item=dtb,                    sunxi.fex
;item=melis-elf,              riscv.fex
;item=logo,                   bootlogo.bmp.lzma
;item=shutdowncharge,         bempty.bmp.lzma
;item=androidcharge,          battery_charge.bmp.lzma
```

tina-dev/device/config/chips/r853/configs/default/boot_package.cfg
默认没有开rsicv

图 4-24: 4.6.3-9riscv 开关

测试结论：

1）该吞吐问题，与 riscv 核无关联。

测试十：关闭了 XR829 驱动的 PROC 和 DEBUG 宏、关闭部分多余的 debug 代码、修改 block size。

吞吐在 68~69Mbps，偶尔到 70、71Mbps

根本原因：单核 CPU 吞吐性能问题

解决办法：

1. 提升频率

2. 提高线程优先级

3. 关闭了 XR829 驱动的 PROC 和 DEBUG 宏、关闭部分多余的 debug 代码，修改 block size。

思路总结：

1. 先检查硬件

1）优先确保硬件 RF 指标正常，查看板卡的 RF 测试报告。

2）确保使用的板卡硬件正常，比如：板载天线是否有接通路电阻，外置天线是否有接，sdio 上下拉电阻是否 OK。

原理图是否有经过审核，尤其是涉及电平转换的电路。

2. 软件问题细化

1）用控制变量法分析，排除单一板卡差异，路由兼容性问题，网络访问控制差异，环境干扰影响等。

2）环境对比测试【区分吞吐性能瓶颈和抗干扰问题】

办公环境，屏蔽房

3）用对比分析法，寻找竞品或者前后版本做对比分析【区分吞吐性能不符合预期还是瓶颈深度优化】

3. 常用优化措施

1）提高 cpu 频率，采用多核，较少 CPU 占用率，减少系统总负荷。。

2）提高线程优先级，线程合并。

3）调整 io 的时钟，例如：io 口的 debounce。

4）修改中断，sdio 中断，gpio 中断。

5）sdio 提频，耐压值处理。

6）驱动速率调整，聚合调整，带宽调整，线程调整。

7）iperf 应用优化。

4. 常用方法

1）对比分析。

2）流程分析。

3）抓大头优化。

信息反馈:

按照排查指南仍无法解决，给出当前初步分析定位的点，提供如下信息：

- 可复现环境

4.7 应用问题

4.7.1 排查思路

一、软件速查

1）采用对比方法，本地测试分析。

【wifimanager/wpa_cli 以及客户的应用进行对比分析】

2）流程分析，从业务逻辑，流程控制梳理问题操作的流程。

【按照云服务，客户应用，中间件接口，驱动，firmware 的层次分析】缩小定位。

4.7.2 案例一：应用调用未触发导致回连失败

问题描述：路由器断电，音箱无法回连。

问题背景：

产品：音箱

主控：R328

模组：XR829

wlan 版本：

```
root@TinaLinux:/# cat sys/kernel/debug/ieee80211/phy0/xradio/version
Driver Label:XR_V02.16.91 _HT40_01.38_BG  Aug  5 2022 01:36:58
Firmware Label:XR_C09.08.52.75_DBG_02.141 2GHZ HT40 Jun 14 2022 09:19:08
```

问题表现：路由器断电重新上电，音箱设备高概率无法回连。

问题分析：

1. 现象确认

应用没有触发回连，没有调用到 wifimanager 的 connect 接口。

[第二次没有触发回连时-路由上电的打印.txt]

路由断电，应用提示连接失败。

```
event name:DISCONNECTED
Network disconnected!
event_label:6
--->WMG_EVENT: WSE_AUTO_DISCONNECTED
--->WMG_STATE: DISCONNECTED
event_label 6, StaEvt.state:4
ssid: hanshow, bssid:
```

网络断开

```
Disconnected,the reason:WSE_AUTO_DISCONNECTED
```

用户主动断连

应用收到 disconnect 事件，发起回连的动作

```
2022- 7-29 14:16:59.270
 /uaibot( 2132:3033644308) DEBUG  [app][OnDisConnected] OnDisConnected ssid: hanshow, bssid:  4 :
        getWifiConfigStatus
2022- 7-29 14:16:59.271
 /uaibot( 2132:3033644308) DEBUG  [app][dispatchMsg] dispatch msg id: 81, name:led effect(714), delay:0s, sender:Sys,
        receiver:Led
2022- 7-29 14:16:59.271
 /uaibot( 2132:3033644308) DEBUG  [app][startConnect] connectState_ : 0
2022- 7-29 14:16:59.271
```

```
/uaibot( 2132:3033644308) INFO  [sdk][Nodelite][stop] DWakeup stop
2022- 7-29 14:16:59.271
/uaibot( 2132:3042901268) DEBUG  [app][dispatchLoop] handle msg id: 81
2022- 7-29 14:16:59.272
/uaibot( 2132:3042901268) DEBUG  [app][HandleMessage] enqueue led msg: 4, index: 23,in=0,isResumeBindDeviceFlag
    =1,isResumeGeekFlag=0
2022- 7-29 14:16:59.272
/uaibot( 2132:3042901268) DEBUG  [app][dispatchLoop] handle msg id: 81 over
2022- 7-29 14:16:59.272
/uaibot( 2132:3005955348) DEBUG  [app][connect] ssid = hanshow, password = 12345678
2022- 7-29 14:16:59.272
/uaibot( 2132:3005955348) DEBUG  [app][connect] Start reset wpa supplicant.
2022- 7-29 14:16:59.273
/uaibot( 2132:3036814612) DEBUG  [app][handleMsg] handle led msg: 4, index: 23
Successfully initialized wpa_supplicant
```

Wpa_supplicant 重启成功，但是后续没有继续调用 connect 接口。

2. 分析应用层网络处理逻辑：

1）断网触发 OnDisConnected() 事件；

音箱收到断网回调或连接错误后开始连接，调用 startConnect()

```
class VboxWifiConnectObserver : public WifiConnectObserver
{
public:
    VboxWifiConnectObserver(WifiManager *wifi): wifiManager_(wifi) {}

    virtual void OnConnectError(int errCode, const string &ssid, const string &bss
    {
        LOG_DEBUG("%s errCode: %d, ssid: %s, bssid: %s", __FUNCTION__, errCode, ss
        GLOBAL->setWifiIsConnecting(false);

        MyWifiControler::GetInstance()->startConnect();
    }
```

图 4-25: 4.7.2-1 应用联网

```
virtual void OnDisConnected(const string &ssid, const string &bssid)
{
    LOG_DEBUG("%s ssid: %s, bssid: %s %d : getWifiConfigStatus", __FUNCTION__, ssid.c_str(),
    if (GLOBAL->getWifiConfigStatus() != WifiConfigState::WIFI_CONFIGURING && !GLOBAL->getWi
    {
        MyWifiControler::GetInstance()->setConnectState(Disabled);
        if (!GLOBAL->muted())
        {
            Dispatcher->dispatchMsg(0, ENTITY_ID_SYS, ENTITY_ID_LED, UAIBOT_MSG_LED_WIFI_DIS
        }
    }
    MyWifiControler::GetInstance()->startConnect();
    VboxWakeup::instance()->stopIDWakeup(); //网络断开-调用分布式的stop
```

图 4-26: 4.7.2-2 应用联网

2）应用开始连接网络；startConnect() 接口是应用封装的, 开启线程操作调用 connect 接口，因为联网是耗时操作。

```
void MyWifiControler::startConnect()        ftshen, 14个月前 • add uaibot源码 ...
{
    LOG_DEBUG("connectState_ : %d ", connectState_);
    if (GLOBAL->getWifiIsConnecting())
    {
        LOG_DEBUG("startConnect GLOBAL->getWifiIsConnecting true ");
        return;
    }
    if (connectState_ == Connecting || connectState_ == Connected) {
        LOG_DEBUG("reconnect thread is run already or connected, return directly.
        return;
    }

    OS_CreateThread((THREAD_PROC)connect, this, E_DETACH_THREAD, 0);
}
```

图 4-27: 4.7.2-3 应用联网

3）这一层 connect 接口仍然是应用封装的。

```
void *MyWifiControler::connect(void *p)
{
    getWifiInfo();
    LOG_DEBUG("ssid = %s, password = %s", ssid_.c_str(), password_.c_str());
    if (ssid_ == "" || password_ == "") {
        LOG_DEBUG("ssid or password is null, return directly.");
        return 0;
    }

    connectState_ = Connecting;
    int err = -1;
    int repeatTimes = 0;

    while(connectState_ != Connected && !isOnLink){
        LOG_DEBUG("Start reset wpa supplicant.");
        system("killall -9 wpa_supplicant");
        system("ifconfig wlan0 up");
        system("cp /etc/wifi/wpa_supplicant.conf /tmp/");
        system("cp /etc/wifi/wpa_supplicant_overlay.conf /tmp/");
        system("wpa_supplicant -iwlan0 -Dnl80211 -c/tmp/wpa_supplicant.conf -I/tmp/wpa_supplicant_overlay.conf -O/etc/wifi/sockets -B");
        sleep(1);

        err = WifiConnector::Instance()->Connect(ssid_, bssid_, password_);
        sleep(5);
        LOG_DEBUG("wifi reconnect times, result code : %d", err);
    }
    connectState_ = Idle;

    return 0;
```

图 4-28: 4.7.2-4 应用联网

先 kill 掉 wpa_supplicant, 再重启 wlan0，然后将需要的配置信息文件拷贝到/tmp 目录，再重启 wpa_supplicant，然后调用 Connect 接口。

4）Connect 接口仍然是应用封装的。

```
int WifiConnector::Connect(const string &ssid, const string &bssid, const string &password, int authType)
{
    auto ret = WifiOn();
    if(ret == -1)  return ret;

    _connect_id = _eventLabel++;

    if (containChinese(ssid)) {
        vector<unsigned char> utf8Codes = encode(ssid);
        unsigned char codesArray[utf8Codes.size()+1] = {0};
        for (int i = 0; i < utf8Codes.size(); i++) {
            printf("0x%X ", utf8Codes[i]);
            codesArray[i] = utf8Codes[i];
        }
        codesArray[utf8Codes.size()] = '\0';
        printf("codesArray: %s\n", codesArray);
        string codesStr = (char *)codesArray;
        _connect_id_ssid_map.insert(make_pair(_connect_id, codesStr));
        _connect_id_bssid_map.insert(make_pair(_connect_id, bssid));
        _pWifiInterface->connect_ap(codesStr.c_str(), password.c_str(), _eventLabel);
    } else {
        _connect_id_ssid_map.insert(make_pair(_connect_id, ssid));
        _connect_id_bssid_map.insert(make_pair(_connect_id, bssid));
        _pWifiInterface->connect_ap(ssid.c_str(), password.c_str(), _eventLabel);
    }


    return 0;
}
```

图 4-29: 4.7.2-5 应用联网

5）这里的 WifiOn() 也是应用封装的，调用 aw_wifi_off()、aw_wifi_on()

```
221  int WifiConnector::WifiOn() {
222      if(_pWifiInterface != NULL) {
223          WifiOff();
224      }
225      printf("Open wifi (aw_wifi_on) ");
226      _pWifiInterface = aw_wifi_on(wifi_state_handle, 0x7fffffff);
227      if(_pWifiInterface == NULL) {
228          printf("failed!!!\n");
229          return -1;
230      }
231      printf("success.\n");
232      return 0;
233  }
234
235  int WifiConnector::WifiOff() {
236      if(_pWifiInterface != NULL) {
237          int ret = aw_wifi_off(_pWifiInterface);        ftshen, 2 years ago • Revert "Revert "
238          if(ret == 0) {
239              _pWifiInterface = NULL;
240              printf("Wifi off success.!!!\n");
241              return 0;
242          } else {
243              printf("Wifi off fail.!!!\n");
244              return -1;
245          }
246      } else {
247          printf("Wifi is already off.\n");
248      }
249      return 0;
250  }
```

图 4-30: 4.7.2-6 应用联网

6）调用 connect_ap() 日志分析如下：

```
24120 Disconnected,the reason:WSE_NETWORK_NOT_EXIST
24121 ap不存在
24122 2022- 7-21 11:02:35.382
24123  /uaibot( 2439:2991619348) DEBUG  [app][OnConnectError] OnConnectError errCode: 2, ssid: , bssid:
24124 2022- 7-21 11:02:35.382
24125  /uaibot( 2439:2991619348) DEBUG  [app][startConnect] connectState_ : 2
24126 2022- 7-21 11:02:35.383
24127  /uaibot( 2439:2991619348) DEBUG  [app][startConnect] reconnect thread is run already or connected, return directly. connectState_ : 2
24128 sendto: Network unreachable
24129 2022- 7-21 11:02:40.383
24130  /uaibot( 2439:2991619348) DEBUG  [app][connect] wifi reconnect times, result code : 0
24131 2022- 7-21 11:02:40.383
24132  /uaibot( 2439:2991619348) DEBUG  [app][connect] Start reset wpa supplicant.
24133 Successfully initialized wpa_supplicant
24134 tang icmp recvfrom:: Resource temporarily unavailable
24135 level: 4
24136 [SD][I][2480][Jul 15 2022 18:14:06][netlink.cpp:181] int check_status(char*, char*, int, int, int):======Network down clear ip=======
24137
24138 2022- 7-21 11:02:41.552
24139  /uaibot( 2439:3017669908) DEBUG  [SD][I][2480][Jul 15 2022 18:14:06][netlink.cpp:181] int check_status(char*, char*, int, int,
      int):======Network down clear ip=======
24140
24141 Command failed: Not found
24142 aw wifi off success!
24143 Wifi off success.!!!
24144 Open wifi (aw wifi_on) success.
24145 0x52 0x65 0x64 0x6D 0x69 0x5F 0x48 0x61 0x69 0x65 0x72 0xE6 0x8E 0x89 0xE7 0xBA 0xBF codesArray: Redmi_Haier掉线
24146 event_label 203, StaEvt.state:2
24147 ssid: Redmi_Haier掉线, bssid:
24148 Connecting to the network......
24149 level: 4
24150 [SD][I][2543][Jul 15 2022 18:14:04][smart_device_manager_impl.cpp:1474] int SmartDeviceManagerImpl2::Impl::dispatch_dev_cloud_state(int):[IN]
      state:299.
```

收到连接失败回调

重启wpa

wifi 关闭成功，打开成功

```
24187 pairs.key [aloST], pairs.value [4]
24188 pairs.key [ssid], pairs.value []
24189 pairs.key [ap_mac], pairs.value []
24190 pairs.key [router_mac], pairs.value []
24191
24192
24193 ret [0]
24194 event_label 203, StaEvt.state:4
24195 ssid: Redmi_Haier掉线, bssid:
24196 网络断开
24197 Disconnected,the reason:WSE_NETWORK_NOT_EXIST
24198 ap不存在
24199 2022- 7-21 11:02:49.240
24200  /uaibot( 2439:2991619348) DEBUG  [app][OnConnectError] OnConnectError errCode: 2, ssid: , bssid:
24201 2022- 7-21 11:02:49.240
24202  /uaibot( 2439:2991619348) DEBUG  [app][startConnect] connectState_ : 2
24203 2022- 7-21 11:02:49.240
24204  /uaibot( 2439:2991619348) DEBUG  [app][startConnect] reconnect thread is run already or connected, return directly. connectState_ : 2
24205 level: 4
24206 [SD][I][2543][Jul 15 2022 18:14:04][smart_device_manager_impl.cpp:1474] int SmartDeviceManagerImpl2::Impl::dispatch_dev_cloud_state(int):[IN]
      state:299.
24207 2022- 7-21 11:02:52.281
24208  /uaibot( 2439:3006749972) DEBUG  [SD][I][2543][Jul 15 2022 18:14:04][smart_device_manager_impl.cpp:1474] int
      SmartDeviceManagerImpl2::Impl::dispatch_dev_cloud_state(int):[IN] state:299.
```

接到回调，再次触发重连

图 4-31: 4.7.2-7 日志分析

图 4-32: 4.7.2-8 日志分析

此时系统报错如下，无法成功连接。

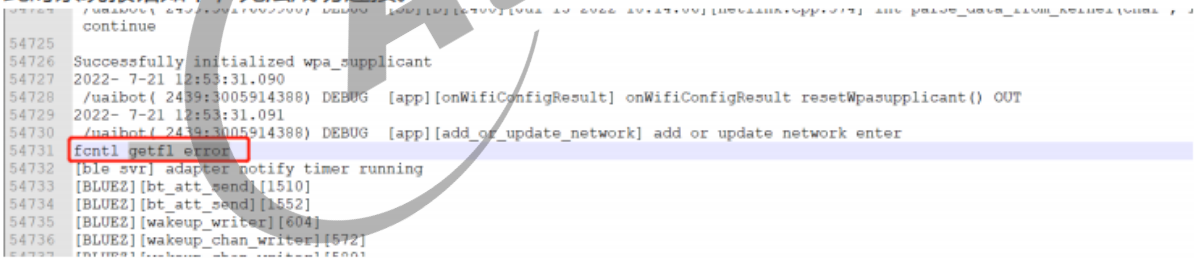

图 4-33: 4.7.2-9 日志分析

3. 流程分析

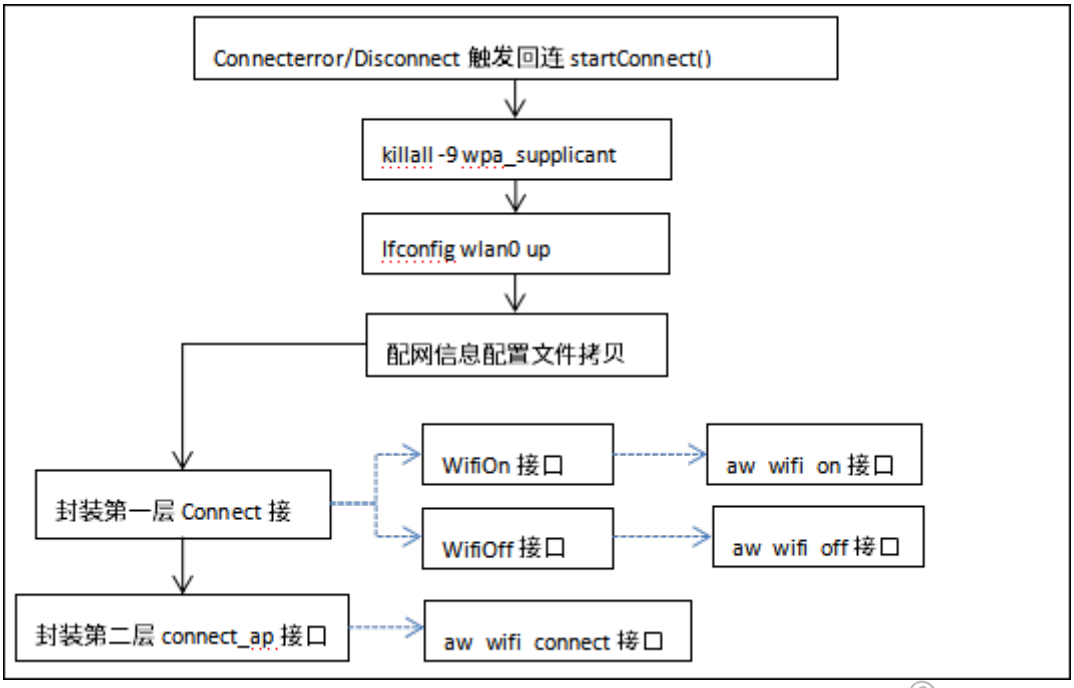

图 4-34: 4.7.2-10 流程图

根本原因：应用不会触发 aw_wifi_connect 接口回调。

解决办法：应用检测网络状态，断网后主动调用一次 aw_wifi_connect 接口。

思路总结：

1. 确认问题背景和现象。

2. 分析并确认从应用到底层驱动的调用逻辑。

3. 猜想并添加日志跟踪定位分析，缩小范围。

4. 精准定位并给出解决方案，找不到根因的优先临时方案，解决问题为先。

信息反馈：

按照排查指南仍无法解决，给出当前初步分析定位的点，提供如下信息：

- 可复现环境
- 分析流程的日志。

4.7.3 案例二：应用没有容错机制导致配网失败

问题描述：弱网环境，空调配网成功率低。

问题背景：

产品：空调

主控：R328

模组：XR829

wlan 版本：

```
root@TinaLinux:/# cat sys/kernel/debug/ieee80211/phy0/xradio/version
Driver Label:XR_V02.16.91 _HT40_01.38_BG  Aug  5 2022 01:36:58
Firmware Label:XR_C09.08.52.75_DBG_02.141 2GHZ HT40 Jun 14 2022 09:19:08
```

问题表现：环境 (-70dB)，反复用手机 app（海尔智家）给空调设备配网，配网成功率低，仅有 23%。

问题分析：

1. 现象确认弱网环境下回概率的配网失败，具体表现 app 显示超时，此时通过日志查看，底层也没有连上网络。2. 底层分析 2.1 针对底层没有联网的现象：咨询客户后了解到他们不会用到蓝牙 mesh，所以释放了蓝牙非 mesh 固件，同时将 wifi 的 fw 也更新到 141 版本，配网成功率提升到 80%。此时表现为：配网时 app 显示超时，但底层实际已经配网成功。2.2 统计配网各阶段耗时

	收到密码	开始联网	联网失败	联网成功	连云成功	开始绑定	绑定失败	绑定成功	联网耗时	连云耗时	绑定耗时	备注
正常网络举例	14:25:14	14:25:15		14:25:19	14:25:27	14:25:27		14:25:28	1	8	1	
失败一(15：47)	15:46:46	15:46:46	15:47:20						36			15：47分 模块在绑定第二步播报网络配置失败
失败二(16：10)	16:09:32	16:09:32	16:10:06						34			16：10分 模块在绑定第二步播报网络配置失败
失败三(16：19)	16:18:45	16:18:46	16:19:20						34			16：19分 模块在绑定第二步播报网络配置失败
失败四(16：28)	未收到密码											16：28分 模块在绑定第一步播报网络配置失败
失败五(16：31)	16:30:51	16:30:51	16:31:25						34			16：31分 模块在绑定第二步播报网络配置失败
问题汇总：	失败1235原因为联网超时，失败4原因为未收到密码											

图 4-35: 4.7.3-1 联网耗时统计

从 iot 同事了解到当前 app 测试的超时时间是 90s，而且是一个总的超时时常，从现象看底层联网的时间远远小于 90s。

3. 流程分析

3.1 梳理配网操作流程

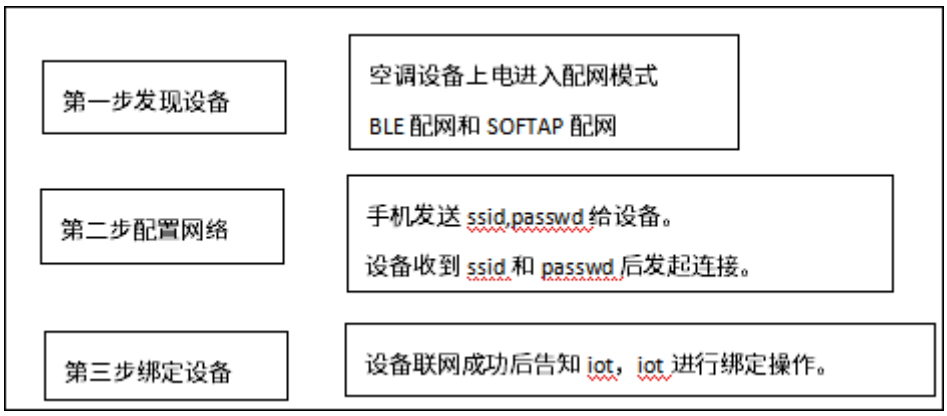

图 4-36: 4.7.3-2 配网流程

3.2 拆解第二步操作

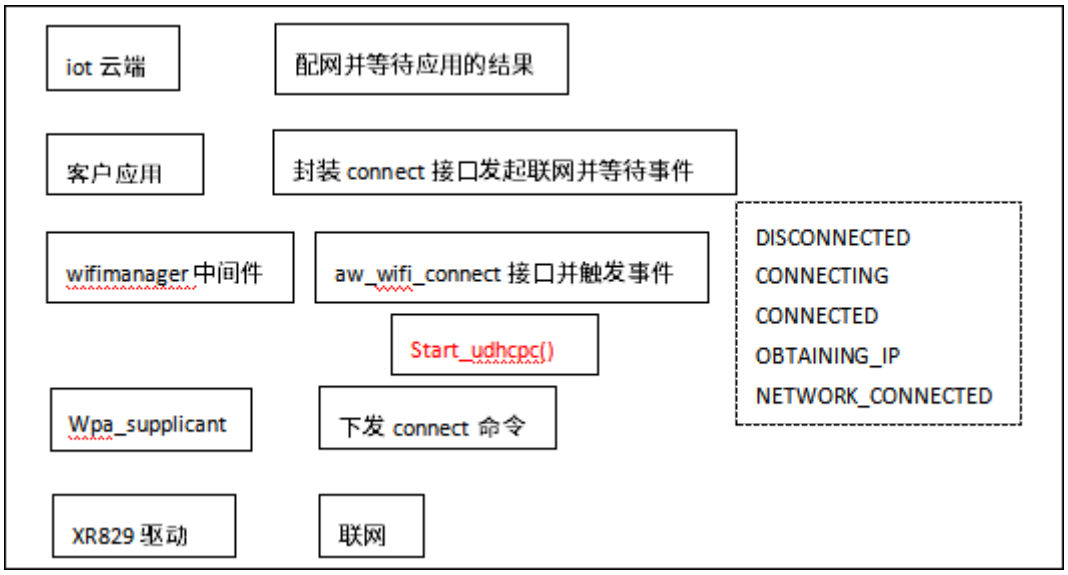

图 4-37: 4.7.3-3 配网流程

3.3 添加应用日志，打开 wifimanager 中间件日志分析：

序号	流程	app日志	网络状态
1	播报进入网络配置模式	[SoundManager.cpp:playLocalTip:51][D] : [SoundManager] Ready play local audio enter_wifi_config.mp3.	DISCONNECTED
2	发现设备	==========Device is found==========	DISCONNECTED
3	收到ssid和psk	[SmartDevicePlug.cpp:ugw_plug_config_start_wifi_sta_callback:955][D] : [SmartDevicePlug] Ssid[9] allwinner, pwd[9] 123456789	DISCONNECTED
4	停止softap网络配置	[WifiManager] Now stop softap config SoftapManager->Disable softap [in] SoftapManager->Disable softap [out]	DISCONNECTED
5	启动sta模式准备联网	[WifiManager.cpp:restartWpaSupplicant:210][D] : [WifiManager] Restart wpa supplicant!!!	DISCONNECTED
6	调用connect_ap	[MoWifiMgr.cpp:connect:283][D] : MoWifiMgr->Store ssid & pwd to list, wait connect wifi in order. [SoundManager.cpp:playLocalTip:51][D] : [SoundManager] Ready play local audio recv_pwd_start_connect_wifi.mp3. [MoWifiMgr.cpp:operator():173][D] : MoWifiMgr->User connect ssid: allwinner, password: 123456789, bssid: 36FB5BE43F91 [Start]	DISCONNECTED
7	启动dhcp超时计数	[SmartDevicePlug.cpp:ugw_plug_config_dhcp_callback:987][D] : [SmartDevicePlug] Wait DHCP timeout [Start]	DISCONNECTED
8	播报收到密码正在连接网络	[LocalAudioPlayer.cpp:play:51][D] : [LocalAudioPlayer] play /mnt/UDISK/module/tone/recv_pwd_start_connect_wifi.mp3	DISCONNECTED
9	网络连接中	aw wifi on success! success. Connect ap with ssid(allwinner), pwd(123456789), label 3 event_label:3 --->WMG_EVENT: WSE_ACTIVE_CONNECT --->WMG_STATE: CONNECTING	CONNECTING
10	网络连接路由成功(未获取ip)	--->WMG_EVENT: WSE_ACTIVE_CONNECT --->WMG_STATE: CONNECTED event_label 3, StaEvt.state:5 ssid: allwinner, bssid: 36FB5BE43F91 Connected to the AP	CONNECTED
11	获取ip地址	--->WMG_EVENT: WSE_ACTIVE_OBTAINED_IP --->WMG_STATE: OBTAINING_IP event_label 3, StaEvt.state:3 ssid: allwinner, bssid: 36FB5BE43F91 Getting ip address..... OBTAINING IPv4.....	OBTAINING_IP

图 4-38: 4.7.3-4 配网流程

12	获取ip地址成功	OBTAINING IPv4 times:0..... event_label:3 --->WMG_EVENT: WSE_ACTIVE_OBTAINED_IP --->WMG_STATE: NETWORK_CONNECTED event_label 3, StaEvt.state:1 ssid: allwinner, bssid: 36FB5BE43F91 联网成功 Successful network connection	NETWORK_CONNECTED
13	更新ip地址成功	Updating IP Address... local IP address: <192.168.204.100> broadcast IP address: <192.168.204.255> Update IP Address Complete 192.168.204.100 192.168.204.255 [08-02 22:28:00::683208][MoWifiObserver.cpp:OnConnected:28][D] : MoWifiObserver->Network connected, ssid : allwinner, bssid : 36FB5BE43F91 [08-02 22:28:00::683396][MoWifiMgr.cpp:notify:301][D] : MoWifiMgr->Wifi status is 0	NETWORK_CONNECTED
14	底层连接路由结束	do cmd STATUS status info: bssid=36:fb:5b:e4:3f:91 freq=2412 ssid=allwinner id=0 mode=station pairwise_cipher=CCMP group_cipher=CCMP key_mgmt=WPA2-PSK wpa_state=COMPLETED ip_address=192.168.204.100 address=84:26:7a:3e:8a:78	NETWORK_CONNECTED
15		[MoWifiMgr.cpp:notify:319][E] : MoWifiMgr->Query rssi fail, set 100 [WifiManager] Wifi connected, auto: true, rssi 100	
16		[ScheduleModule.cpp:loadSchedulesFromCloud:225][D] : [ScheduleModule] load schedules from cloud url http://ai.haier.net:11000/access/ai-access/dot/schedule/query [ScheduleModule.cpp:loadSchedulesFromCloud:248][D] : [ScheduleModule] Load cloud schedule data: {"data":[],"processorContextList":[],"response":"小i你还没有设置日程，需要定日程可以对我说定个日程","retCode":"00000","retInfo":"小i操作成功","sn":"20220802223002910000241986","triggerId":"","uSpanId":"0.1","uTraceId":"0280b93a11fa4cf7a1c7bb1785554483"}	
17	准备播报网络配置失败	[SoundManager.cpp:playLocalTip:51][D] : [SoundManager] Ready play local audio wifi_config_failed.mp3.	
18	停止网络配置准备触发重连	[SmartDevicePlug.cpp:stopWifiConfigMode:1144][D] : [SmartDevicePlug] Need restart wpa supplicant. [WifiManager.cpp:restartWpaSupplicant:210][D] : [WifiManager] Restart wpa supplicant!!!	

图 4-39: 4.7.3-5 配网流程

和失败时的日志对比，找到突破口：WSE_OBTAINED_IP_TIMEOUT

在弱网环境下，的确会概率性的无法获取到 ip 地址（分配 ip 的动作是路由器的行为），wifimanager 当前只能添加 try 的机制，而且从日志打印和实际表现看，最后也的确再次获取到了 ip 地址，也联网成功了。

```
91  int  udhcpc_v4(void)
92  {
93      int times = 0;
94      char ipaddr[NI_MAXHOST] = {0};
95      char cmd[256] = {0}, reply[8] = {0};
96
97      wmg_printf(MSG_DEBUG,"OBTAINING IPv4......\n");
98      /* restart udhcpc */
99      system("/etc/wifi/udhcpc_wlan0 start >/dev/null");
100
101     /* check ip exist */
102     ms_sleep(1000);
103     times = 0;
104     do{
105         ms_sleep(500);
106         get_net_ip("wlan0", AF_INET, ipaddr, NI_MAXHOST);
107         wmg_printf(MSG_DEBUG,"OBTAINING IPv4 times:%d......\n", times);
108         times++;
109     }while((ipaddr[0] == 0) && (times < 10));
110
111     if(ipaddr[0] != 0)
112         return 0;
113     else
114         return -1;
115 }
```

图 4-40: 4.7.3-6dhcp

当前代码命名就有 try 的机制，底层已经成功获取到了 ip 地址，但还是失败，猜想应用没有及时将

状态正确反馈到云端。

追踪客户的代码流程发现，上层应用再调用 connect 时目前没有容错机制，只要失败了，就会立刻将结果返回给 iot，iot 判定未联网，所有 app 就一直等待，最后超时。即使底层再次 dhcp 成功，配网成功，后面的结果没有上报给 iot。

根本原因：

弱网环境，会存在 dhcp 失败，导致联网耗时加长，甚至联网失败。

应用处理的逻辑没有加容错机制，即使底层有容错机制，后面成功的结果，应用没有上报给 iot，iot 拿到的是第一次失败的结果，所有一直认为是超时失败。

解决办法：

应用添加一次容错机制，第一次失败即：收到 WSE_OBTAINED_IP_TIMEOUT 先不上报给 iot，再次等待一次，这里设置一个超时，如果 60s 还没有成功，再将失败结果上报给 iot。

思路总结：

1. 确认问题背景和现象。

2. 分析并确认从应用到底层驱动的调用逻辑。

3. 猜想并添加日志跟踪，对比分析，定位分析，缩小范围。

4. 精准定位并给出解决方案，找不到根因的优先临时方案，解决问题为先。

5. 涉及跨多业务流程的问题，首要的就是先分析流程：

操作流程，业务流程，调用流程【重点要搞清楚输入和输出：参数和返回值，尤其是对返回值的处理。】

信息反馈:

按照排查指南仍无法解决，给出当前初步分析定位的点，提供如下信息：

- 可复现环境
- 分析流程的日志。
- 应用调用的详细逻辑。

著作权声明

商标声明

免责声明