# target/linux/ramips/image目录

## 说明

该目录下是生成固件的信息文件和Makefile

## 文件介绍

## Makefile

## mt7620.mk mt7621.mk mt76x8.mk rt288x.mk rt305x-legacy.mk rt305x.mk rt3883.mk

包含了某个芯片下的路由器产品，每个产品都定义为类似如下格式：

define Device/psg1218b

DTS := PSG1218B

DEVICE\_TITLE := Phicomm PSG1218 rev.Ax

DEVICE\_PACKAGES := kmod-mt76x2

SUPPORTED\_DEVICES += psg1218

endef

TARGET\_DEVICES += psg1218a

TARGET\_DEVICES参数的值就是生成固件后的固件名称：openwrt-ramips-mt7620-psg1218b-squashfs-sysupgrade.bin

烧录路由器后，在/etc/board.json文件中

{

"model": {

"id": "psg1218a",

"name": "Phicomm PSG1218 rev.A"

},

"led": {

"Network": {

Id参数与之对应,在做固件升级的时候，可以通过读取路由器目录/etc下的board.json文件中的id参数 来获取到目标板对应的固件

# package/目录

## 说明

该目录是公共原始配置文件目录，包括了防火墙、wifi、系统配置等等

修改该目录文件，会编译打包成固件

# 开机初始化脚本：config\_generate

## 说明

此文件是在开机的时候运行的配置文件，比如etc/config下的配置文件都是由此脚本生成

package/base-files/files/bin

# 设备的一些信息Borad.json

## 说明

该json文件是记录了设备的一些属性信息，包括网络设备的一些信息

config\_generate就是通过读取该json文件来进行对设备的配置以及生成相关配置文件

# 密码登录配置

### 修改root密码

修改位置：source/package/base-files/files/etc/shadow ------- DebugAdmin123

root:$1$m2FG5gWj$2pFVvsv5pYVuRYUe85BNR.:0:0:99999:7:::

会出现breed刷机后第一次无法登录的情况，需要重新再用breed恢复出厂设置就可以登录了

### 修改inittab文件

不用修改

### menuconfig配置

Base system --->

<\*> busybox ......

[\*] Customize busybox option

Login/Password Management Utilities --->

[\*] login (NEW)

# SSH配置

## Menuconfig配置

Network --->

<\*> SSH ......

<\*> openssh-sftp-server.................................. OpenSSH SFTP server

# 配置WIFI

## 修改脚本文件，该脚本分析了网卡信息

修改位置：source/package/kernel/mac80211/files/lib/wifi/mac80211.sh

set wireless.radio${devidx}.disabled=0 #改为0显示wifi

set wireless.default\_radio${devidx}=wifi-iface

set wireless.default\_radio${devidx}.device=radio${devidx}

set wireless.default\_radio${devidx}.network=lan

set wireless.default\_radio${devidx}.mode=ap

set wireless.default\_radio${devidx}.ssid=LEDE #wifi名称

set wireless.default\_radio${devidx}.encryption=psk2 #wifi加密方式，没有是none

set wireless.default\_radio${devidx}.key=password #wifi密码

# 编译固件出现too big的问题

## 解决方法

该问题主要出现在4M的固件，我们前辰的固件目前无法支持4Mflash的路由器，所以遇到此类问题要么是修改大小，要么就是在屏蔽生成代码，不生成4M的固件，修改文件都在target/linux/rampis/image/

以编译MT7620系列为例，在target/linux/rampis/image/ 打开mt7620.mk，知道ex2700型号，有一个字段是ex2700\_mtd\_size=3866624,把3866624（3MB）改大

# 内存修改

## Ar9344

ar9344支持自动识别内存大小，不需要修改指定大小

## MT76\*\*

在 target/linux/ramips/dts/下找到相应的dts文件，从中查找到mem相关字段进行修改设置

# 前辰固件所需依赖库

## Menuconfig 🡪 Libraries

### Openssl

### Libevent2

### libpcap

### Libevent-openssl

### Json

## Menuconfig 🡪 Development 🡪 Libraries

### Zlib-dev

# 编译库后打包固件会出现缺失lib问题

## 问题描述

在编译固件过程中出现了如下问题：

**Package** base-files is missing dependencies for the following libraries:

libcrypto.so.1.0.0

libevent-2.0.so.5

libevent\_openssl-2.0.so.5

libssl.so.1.0.0

libz.so.1

## 解决方法

在相应的Makefile中（Makefile就是上述错误的Package base-files，意思就是在package目录下的base-files目录下的Makefile）追加DEPENDS：

+libevent2 +libevent2-openssl +zlib

# 前辰所需编译出来的库路径

## 库路径

staging\_dir/<target-\*\*\* target路径>/usr/lib

例如mt7620的路径: staging\_dir/target-mipsel\_24kc\_musl-1.1.16/usr/lib

<target-\*\*\* target路径> 根据不同芯片方案编译出来的路径不一样，找对应的路径即可

## Include路径

staging\_dir/<target-\*\*\* target路径>/usr/lib

例如mt7620的路径: staging\_dir/target-mipsel\_24kc\_musl-1.1.16/usr/include

<target-\*\*\* target路径> 根据不同芯片方案编译出来的路径不一样，找对应的路径即可

# 所需命令

## Iptables 默认自带

## Ipset 需要加入

Network -🡪

<\*> ipset....................................... IPset administration utility

## Tc 需要加入

Network -🡪

Routing and Redirection --->

<\*> tc............................................... Traffic control utility

## Ip 默认自带

# Kmod-ifb支持

Kernel modules -🡪

Network Devices -----🡪

<\*> kmod-ifb........................... Intermediate Functional Block support

# 时区设置

## 修改时间和国家

在目录source/package/base-files/files/bin/config\_generate

修改config\_generate配置文件

generate\_static\_system() {

json\_select modle 🡨----------------此处源码有错误，实际是model

json\_get\_vars name

hostname=$name

if [ -z "$name" ]; then

hostname="QianChen"

fi

echo "$hostname" > /etc/app/model

uci -q batch <<-EOF

delete system.@system[0]

add system system

set system.@system[-1].hostname='$hostname'

set system.@system[-1].timezone='CST-8'

set system.@system[-1].zonename='Asia/Shanghai'

set system.@system[-1].ttylogin='0'

set system.@system[-1].log\_size='64'

set system.@system[-1].urandom\_seed='0'

# 刷机方法

## Uboot命令行刷机

## Web页面刷机

如果用web刷那么选择 带有 factory的bin,再刷sysupgrade

## 通过命令sysupgrade刷机

Ssh到设备终端，把对应设备的sysupgrade.bin固件下载到路由器/tmp目录下，通过sysupgrade –v \*\*\*\*.bin

## 通过breed刷机

路由器上电的同时按住reset按键大约10秒钟，之后路由器lan口连接电脑网口，登录web页面 192.168.1.1 就可以进入breed刷机页面

1. 环境变量 禁用
2. 恢复出厂设置
3. 固件更新，选择固件

# 获取客户端信息

## 通过iw获取详细信息

### 获取所有客户端信息

iw dev wlan0 station dump

Station 50:82:d5:a2:eb:9c (on wlan0)

inactive time: 1540 ms

rx bytes: 19380 #流量大小

rx packets: 216

tx bytes: 15020

tx packets: 92

tx retries: 0

tx failed: 1

rx drop misc: 1

signal: -36 [-50, -36] dBm

signal avg: -40 [-53, -40] dBm

tx bitrate: 65.0 MBit/s VHT-MCS 1 80MHz short GI VHT-NSS 1

rx bitrate: 24.0 MBit/s

expected throughput: 31.218Mbps

authorized: yes

authenticated: yes

associated: yes

preamble: long

WMM/WME: yes

MFP: no

TDLS peer: no

DTIM period: 2

beacon interval:100

short slot time:yes

connected time: 15 seconds

### 获取指定客户端信息

iw dev wlan0 station get <客户端MAC>

iw dev wlan0 station get 50:82:d5:a2:eb:9c

Station 50:82:d5:a2:eb:9c (on wlan0)

inactive time: 19760 ms

rx bytes: 79705

rx packets: 676

tx bytes: 31481

tx packets: 209

tx retries: 28

tx failed: 1

rx drop misc: 3

signal: -37 [-37, -42] dBm

signal avg: -39 [-39, -44] dBm

tx bitrate: 180.0 MBit/s VHT-MCS 8 40MHz short GI VHT-NSS 1

rx bitrate: 24.0 MBit/s

expected throughput: 60.333Mbps

authorized: yes

authenticated: yes

associated: yes

preamble: long

WMM/WME: yes

MFP: no

TDLS peer: no

DTIM period: 2

beacon interval:100

short slot time:yes

connected time: 88 seconds

## 获取wifi连接数量

iw dev wlan0 station dump |grep Station |wc -l #wifi连接人数

## 连接的用户MAC

iw dev wlan0 station dump |grep Station |awk -F ' ' ' {print $2}' #连接的用户MAC

## 连接用户的信号强弱

iw dev wlan0 station dump #连接用户的信号强弱

## 查看wifi信息

iw wlan0 info #查看wifi信息

## 注意

客户端断开连接后，iw查询会有延时大约5-8分钟时间才能更新状态

/proc/net/arp 下的更新延迟会更久，实际项目中建议iw与/proc/net/arp结合起来使用

## iwinfo 响应可以更快速的获取连接状态

iwinfo <wifi device> assoclist

e.g

iwinfo wlan0 assoclist

50:82:D5:A2:EB:9C -60 dBm / unknown (SNR -60) 780 ms ago

RX: 24.0 MBit/s 249 Pkts.

TX: 28.9 MBit/s, MCS 3, 20MHz 108 Pkts.

该iwinfo方法，可以非常实时的查询到客户端连接信息，当客户端断开了wifi连接后，可以即时响应判断到客户端连接状态

# 区分5G和2G

## 通过Wireless配置文件区分

通过source/package/kernel/mac80211/files/lib/wifi/mac80211.sh可以知道，当该脚本判断到5G网卡设备时，会把hwmode赋值的值存在 ’a’，开发者程序中只需读出hwmode是否有’a’，如果存在则为 5G 否则为2G

# 限制wifi接入量

## 修改Wireless配置文件

### MT7620限制方法

config wifi-device 'ra0'

             option type 'rt2860v2'

             option hwmode '11g'

             option channel 'auto'

             option txpower '100'

             option htmode 'HT40'

             option country 'CN'

             option maxassoc '25' #这个就是限制数量

在wireless配置文件中配置wifi-device节点，增加选项：

option maxassoc ‘数量’

对于双网卡（2G和5G），需要分别设置

我们看到maxassoc 这个字段是在wifi-deivce下面的。MT720的wifi连接数限制是对应接口的，  
上面说的MT7620的wifi连接数限制是对应接口的，表示该接口下所有的SSID的wifi用户连接数限制都是这个值, 不是该接口下所有SSID用户相加总数限制

### AR9341/AR9344/AR93\*\*系列限制方法

config wifi-device 'radio0'

             option type 'mac80211'

             option hwmode '11g'

             option path 'platform/ar934x\_wmac'

             option mode '9'

             option country 'US'

             option channel '11'

             option htmode 'HT20'

             option txpower '22'

config wifi-iface

             option device 'radio0'

             option network 'lan'

             option mode 'ap'

             option encryption 'none'

             option ssid 'test'

             option maxassoc '25'

我们看到maxassoc 这个字段是在wifi-iface下面的，这个跟MT7620不一样，AR93系列的wifi连接数限制是对应ssid的，表示这个ssid的wifi用户连接数限制。

# 获取WIFI网卡设备信息

## 获取WIFI网卡的txpower list

iwinfo <wifi device> txpowerlist

e.g

iwinfo wlan0 txpowerlist

0 dBm ( 1 mW)

1 dBm ( 1 mW)

2 dBm ( 1 mW)

3 dBm ( 1 mW)

4 dBm ( 2 mW)

5 dBm ( 3 mW)

6 dBm ( 3 mW)

7 dBm ( 5 mW)

8 dBm ( 6 mW)

9 dBm ( 7 mW)

10 dBm ( 10 mW)

11 dBm ( 12 mW)

12 dBm ( 15 mW)

13 dBm ( 19 mW)

14 dBm ( 25 mW)

15 dBm ( 31 mW)

16 dBm ( 39 mW)

17 dBm ( 50 mW)

18 dBm ( 63 mW)

\* 19 dBm ( 79 mW)

20 dBm ( 100 mW)

21 dBm ( 125 mW)

22 dBm ( 158 mW)

## 获取WIFI频段freqlist与通道对应关系

iwinfo <wifi device> freqlist

restricted: 受限制的

e.g

**5G**

iwinfo wlan0 freqlist

\* 5.180 GHz (Channel 36)

5.200 GHz (Channel 40)

5.220 GHz (Channel 44)

5.240 GHz (Channel 48)

5.260 GHz (Channel 52)

5.280 GHz (Channel 56)

5.300 GHz (Channel 60)

5.320 GHz (Channel 64)

5.500 GHz (Channel 100)

5.520 GHz (Channel 104)

5.540 GHz (Channel 108)

5.560 GHz (Channel 112)

5.580 GHz (Channel 116)

5.600 GHz (Channel 120)

5.620 GHz (Channel 124)

5.640 GHz (Channel 128)

5.660 GHz (Channel 132)

5.680 GHz (Channel 136)

5.700 GHz (Channel 140)

5.745 GHz (Channel 149) [restricted]

5.765 GHz (Channel 153) [restricted]

5.785 GHz (Channel 157) [restricted]

5.805 GHz (Channel 161) [restricted]

5.825 GHz (Channel 165) [restricted]

**2G:**

iwinfo wlan1 freqlist

2.412 GHz (Channel 1)

2.417 GHz (Channel 2)

2.422 GHz (Channel 3)

2.427 GHz (Channel 4)

2.432 GHz (Channel 5)

2.437 GHz (Channel 6)

2.442 GHz (Channel 7)

2.447 GHz (Channel 8)

2.452 GHz (Channel 9)

2.457 GHz (Channel 10)

\* 2.462 GHz (Channel 11)

2.467 GHz (Channel 12) [restricted]

2.472 GHz (Channel 13) [restricted]

2.484 GHz (Channel 14) [restricted]

## 获取assoclist（WIFI客户端信息）

iwinfo <wifi device> assoclist

e.g

iwinfo wlan0 assoclist

50:82:D5:A2:EB:9C -60 dBm / unknown (SNR -60) 780 ms ago

RX: 24.0 MBit/s 249 Pkts.

TX: 28.9 MBit/s, MCS 3, 20MHz 108 Pkts.

该iwinfo方法，可以非常实时的查询到客户端连接信息，当客户端断开了wifi连接后，可以即时响应判断到客户端连接状态

## 获取htmodelist（带宽）

iwinfo <wifi device> assoclist

e.g

iwinfo wlan1 htmodelist

HT20 HT40

# WIFI配置

## 无线配置文件的一些参数

option htmode HT40+

使用40MHz频宽，但只支持1-7信道。HT40-，支持5-13信道。HT20 支持1-13信道，20MHz频宽 option channel 1

使用信道1,1-13之间。若有限定频宽，信道参考如上。 option country CN

设定wifi标准为中国 CN、JP、HK均可开启13信道，默认的US只有1-12信道 option noscan 1 强制使用40MHz的频宽 option txpower 17

发射功率为17dBm，大约50mW。不可设定过大，否则有烧毁PA的危险

# 踢掉用户WIFI连接

## 通过iw实现该行为

iw dev <wifi device> station del <mac>

e.g : iw dev wlan0 station del 50:82:d5:a2:eb:9c

该方法只能踢掉当前连接，用户可以下次重新连接，并不会永久踢出

如果要做到永久踢掉，需要配合iptables来实现

# 基于多SSID的访客控制

## 说明

目标. 从无线网络中设置多个无线SSID，从而控制不同SSID访客权限

## 相关配置文件

package/network/config/firewall/files/firewall.config

package/base-files/files/bin/config\_generate

package/kernel/mac80211/files/lib/wifi/mac80211.sh

package/network/services/dnsmasq/files/dhcp.conf

## 步骤

### 定义新的网络接口

/etc/config/network  
  
config interface 'guest'  
        option type 'bridge'  
        option proto 'static'  
        option ipaddr '10.0.0.1'  
        option netmask '255.255.255.0'  
        option \_orig\_ifname 'radio0.network2'

option \_orig\_bridge 'true'

修改package/base-files/files/bin/config\_generate

guest\_flag=""

addr\_offset=2

generate\_network() {

local ifname macaddr protocol type ipaddr netmask

json\_select network

json\_select "$1"

json\_get\_vars ifname macaddr protocol ipaddr netmask

json\_select ..

json\_select ..

[ -n "$ifname" ] || return

guest\_flag=""

# force bridge for multi-interface devices (and lan)

case "$1:$ifname" in

\*\ \* | lan:\*)

type="bridge"

guest\_flag="1"

;;

esac

uci -q batch <<-EOF

delete network.$1

set network.$1='interface'

set network.$1.type='$type'

set network.$1.ifname='$ifname'

set network.$1.proto='none'

EOF

[ -n "$macaddr" ] && uci -q batch <<-EOF

delete network.$1\_dev

set network.$1\_dev='device'

set network.$1\_dev.name='$ifname'

set network.$1\_dev.macaddr='$macaddr'

EOF

case "$protocol" in

static)

local ipad

case "$1" in

lan) ipad=${ipaddr:-"192.168.1.1"} ;;

\*) ipad=${ipaddr:-"192.168.$((addr\_offset++)).1"} ;;

esac

netm=${netmask:-"255.255.255.0"}

uci -q batch <<-EOF

set network.$1.proto='static'

set network.$1.ipaddr='$ipad'

set network.$1.netmask='$netm'

set network.$1.ip6assign='60'

EOF

;;

dhcp)

# fixup IPv6 slave interface if parent is a bridge

[ "$type" = "bridge" ] && ifname="br-$1"

uci -q batch <<-EOF

set network.$1.proto='dhcp'

delete network.${1}6

set network.${1}6='interface'

set network.${1}6.ifname='$ifname'

set network.${1}6.proto='dhcpv6'

EOF

;;

pppoe)

uci -q batch <<-EOF

set network.$1.proto='pppoe'

set network.$1.username='username'

set network.$1.password='password'

set network.$1.ipv6='1'

delete network.${1}6

set network.${1}6='interface'

set network.${1}6.ifname='@${1}'

set network.${1}6.proto='dhcpv6'

EOF

;;

esac

#guest

if [ "$guest\_flag" == "1" ]; then

uci -q batch <<-EOF

delete network.guest

set network.guest='interface'

set network.guest.type='$type'

set network.guest.proto='static'

set network.guest.ipaddr='10.0.0.1'

set network.guest.netmask='255.255.255.0'

set network.guest.\_orig\_ifname='$ifname.network2'

set network.guest.\_orig\_bridge='true'

EOF

fi

}

### 定义新的无线SSID

/etc/config/wireless

config wifi-iface  
        option device 'radio0'  
        option mode 'ap'  
        option encryption 'none'  
        option ssid ‘Free'  
        option network 'guest'

修改package/kernel/mac80211/files/lib/wifi/mac80211.sh

detect\_mac80211() {

devidx=0

config\_load wireless

while :; do

config\_get type "radio$devidx" type

[ -n "$type" ] || break

devidx=$(($devidx + 1))

done

for \_dev in /sys/class/ieee80211/\*; do

[ -e "$\_dev" ] || continue

dev="${\_dev##\*/}"

found=0

config\_foreach check\_mac80211\_device wifi-device

[ "$found" -gt 0 ] && continue

fre="2g"

mode\_band="ng"

channel="11"

htmode="HT40"

ht\_capab=""

iw phy "$dev" info | grep -q 'Capabilities:' && htmode=HT20

iw phy "$dev" info | grep -q '2412 MHz' || { mode\_band="a"; channel="36"; }

vht\_cap=$(iw phy "$dev" info | grep -c 'VHT Capabilities')

cap\_5ghz=$(iw phy "$dev" info | grep -c "Band 2")

[ "$vht\_cap" -gt 0 -a "$cap\_5ghz" -gt 0 ] && {

mode\_band="na";

channel="161"

htmode="VHT80"

fre="5g"

}

[ -n "$htmode" ] && ht\_capab="set wireless.radio${devidx}.htmode=$htmode"

if [ -x /usr/bin/readlink -a -h /sys/class/ieee80211/${dev} ]; then

path="$(readlink -f /sys/class/ieee80211/${dev}/device)"

else

path=""

fi

if [ -n "$path" ]; then

path="${path##/sys/devices/}"

case "$path" in

platform\*/pci\*) path="${path##platform/}";;

esac

dev\_id="set wireless.radio${devidx}.path='$path'"

else

dev\_id="set wireless.radio${devidx}.macaddr=$(cat /sys/class/ieee80211/${dev}/macaddress)"

fi

ssid\_name="QianChen-${fre}"

guest\_ssid\_name="QianChen-guest-${fre}"

uci -q batch <<-EOF

set wireless.radio${devidx}=wifi-device

set wireless.radio${devidx}.type=mac80211

set wireless.radio${devidx}.channel=${channel}

set wireless.radio${devidx}.hwmode=11${mode\_band}

set wireless.radio${devidx}.country=CN

${dev\_id}

${ht\_capab}

set wireless.radio${devidx}.disabled=0

set wireless.default\_radio${devidx}=wifi-iface

set wireless.default\_radio${devidx}.device=radio${devidx}

set wireless.default\_radio${devidx}.network=lan

set wireless.default\_radio${devidx}.mode=ap

set wireless.default\_radio${devidx}.ssid=${ssid\_name}

set wireless.default\_radio${devidx}.encryption=none

set wireless.guest\_radio${devidx}=wifi-iface

set wireless.guest\_radio${devidx}.device=radio${devidx}

set wireless.guest\_radio${devidx}.network=guest

set wireless.guest\_radio${devidx}.mode=ap

set wireless.guest\_radio${devidx}.ssid=${guest\_ssid\_name}

set wireless.guest\_radio${devidx}.encryption=none

EOF

uci -q commit wireless

devidx=$(($devidx + 1))

done

}

### 增加防火墙规则，让客户网络可以访问外网

/etc/config/firewall  
  
config zone  
        option name 'guest'  
        option input 'ACCEPT'

        option forward 'REJECT'  
        option output 'ACCEPT'  
        option network 'guest'  
  
config forwarding  
        option src 'guest'  
        option dest 'wan'  
  
config rule  
        option name 'DNSGuest'  
        option src 'guest'  
        option dest\_port '53'  
        option proto 'tcpudp'  
        option target 'ACCEPT'  
  
config rule  
        option name 'DHCPGuest'  
        option src 'guest'  
        option src\_port '67-68'  
        option dest\_port '67-68'  
        option proto 'udp'  
        option target 'ACCEPT'

### 定义DHCP

/etc/config/dhcp

config dnsmasq  
        option domainneeded '1'  
        option boguspriv '1'  
        option localise\_queries '1'  
        option rebind\_protection '1'  
        option rebind\_localhost '1'  
        option local '/lan/'  
        option domain 'lan'  
        option expandhosts '1'  
        option readethers '1'  
        option leasefile '/tmp/dhcp.leases'  
        list interface 'lan'  
        list interface 'guest'  
  
  
config dhcp 'lan'  
        option interface 'lan'  
        option leasetime '12h'  
        option start '105'  
        option limit '118'

config dhcp 'guest'  
        option interface 'guest'  
        option leasetime '12h'  
        option start '100'  
        option limit '119'

### 权限控制

利用iptables可以控制新的无线SSID网络

# TC限速

## 基本命令语句

### 删除规则

#### 删除root

tc qdisc del dev wan root

#### 删除ingress

tc qdisc del dev wan handle ffff: ingress

#### 获取class结点的流量大小

tc -s class ls dev eth0.2 classid 1:3

### 流量重定向

tc qdisc add dev wan handle ffff: ingress

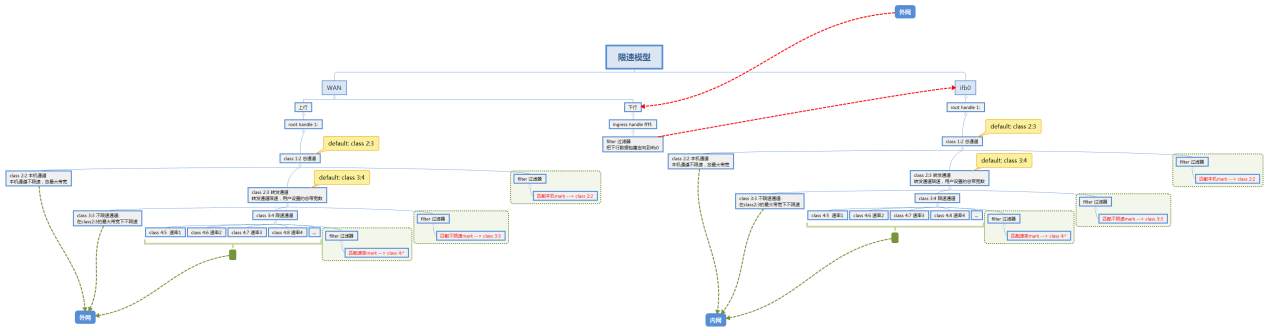
//把下行的流量重定向到虚拟网卡

tc filter add dev wan parent ffff: protocol ip u32 match u32 0 0 action mirred egress redirect dev ifb0

## 限速模型设计

### 模型图要

清晰图看原图



# 固件说明

## 斐讯系列:

openwrt-ramips-mt7620-psg1208-squashfs-sysupgrade.bin：K1

openwrt-ramips-mt7620-psg1218a-squashfs-sysupgrade.bin: 五个网口的K2

openwrt-ramips-mt7620-psg1218b-squashfs-sysupgrade.bin：四个网口的K2C

# MTK wifi出现无法连接的问题

## 问题描述

Mtk7620路由器长时间运行后出现wifi无法连接，从dmesg中看到日志

ieee80211 phy0: rt2x00queue\_write\_tx\_frame: Error - Dropping frame due to full tx queue 2

## 解决方法

### 修改驱动文件 rt2x00queue.c：

build\_dir/target-mipsel\_24kc\_musl/linux-ramips\_mt7620/linux-4.9.82/drivers/net/wireless/ralink/rt2x00/rt2x00queue.c

把int rt2x00queue\_write\_tx\_frame(struct data\_queue \*queue, struct sk\_buff \*skb,

struct ieee80211\_sta \*sta, bool local)函数中

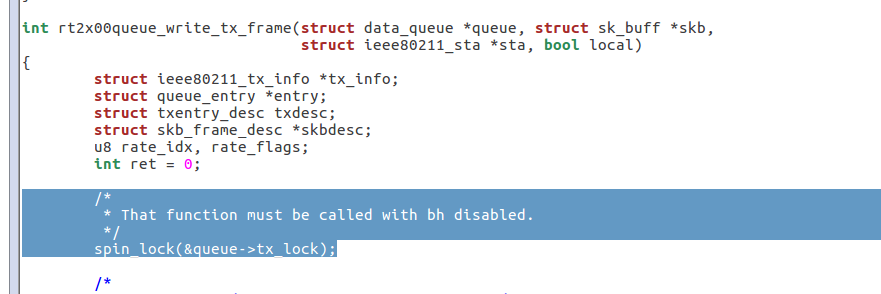
/\*

\* That function must be called with bh disabled.

\*/

spin\_lock(&queue->tx\_lock);

提到函数开始处



### 修改驱动文件rt2800mmio.c:

build\_dir/target-mipsel\_24kc\_musl/linux-ramips\_mt7620/backports-2017-11-01/drivers/net/wireless/ralink/rt2x00/rt2800mmio.c

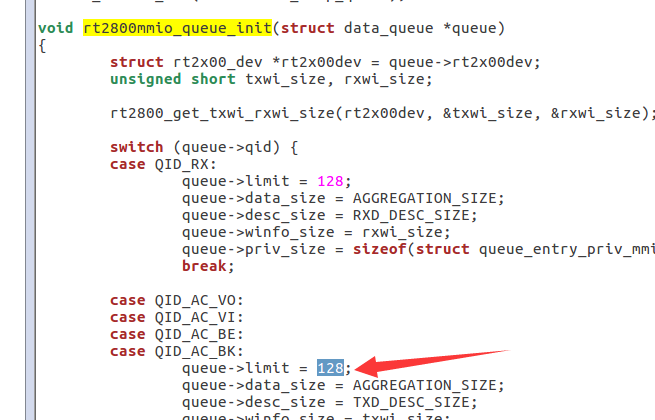
修改尺寸

void rt2800mmio\_queue\_init(struct data\_queue \*queue)中

case QID\_AC\_BE:

case QID\_AC\_BK:

queue->limit = 128;//原来是64



修改完这两个文件后重新编译固件，刷入固件，运行一段时间偶尔会出现这样的提示：

 ieee80211 phy0: rt2800mmio\_txstatus\_is\_spurious: Warning - 4 spurious TX\_FIFO\_STATUS interrupt(s)

 ieee80211 phy0: rt2800mmio\_txdone: Warning - Got TX status for an empty queue 2, dropping

但是却不影响wifi上网，应该也无所谓了。

### 注意

每次git pull更新openwrt源码时，如果更新了内核，上述文件将会覆盖成新的内核文件，需要重新修改上述文件