RT-WiFi User guide v0.1

Contents

Introduction

This user guide is for the release of RT-WiFi driver v0.1. RT-WiFi driver is built on top of the ath9k driver in backports driver collection. This release v0.1 is based on backports driver v3.13.2 (https://www.kernel.org/pub/linux/kernel/projects/backports/stable/v3.13.2/). Currently, RT-WiFi supports IEEE 802.11 a/b/g infrastructure mode. In v0.1, we implement basic TDMA features and static TDMA link schedule. Please go through the following user guide before you use this driver.

System Requirement

We have tested RT-WiFi driver in the following hardware and software configurations.

Hardware Configuration

- x86 32-bit or 64 bit machines.
- Atheros ath9k based Wi-Fi chips. We have tested on AR9280 chip.
  - We recommend Atheros chips that support 5GHz band, since there are less interference in the 5GHz channels.
  - We have tested our driver in IEEE802.11a and IEEE 802.11g.

Software Configuration

- Ubuntu 14.04 32-bit or 64-bit OS.
- Linux kernel 3.13.0.32 or before.

Network Configuration

Before using RT-WiFi driver, please make sure to properly configure the software in both the access point (AP) and the station (STA) side. We recommend to firstly use the original backports driver (https://www.kernel.org/pub/linux/kernel/projects/backports/stable/v3.13.2/) to test the network configuration. In the AP side, we use hostapd to configure wireless AP and to serve authentication process. To dynamically allocate IPs to wireless stations (STAs), we use **isc-dhcp-server**. In the station side, **wpa_supplicant** is used to authenticate with authentication server.

AP side configuration

1. Software installation

```
$ sudo apt-get install iw hostapd isc-dhcp-server
```

2. Make sure the WLAN card supports AP mode

```
$ iw list
```

3. Refer the following configuration files to configure hostapd and isc-dhcp-server.

hostapd configuration

After installing hostapd, refer to the following sample configuration to set hostapd.

```
#Sample configuration for hostapd
#/etc/hostapd/hostapd.conf

#Make sure you set the right interface
interface=wlan1
driver=nl80211

logger_syslog=-1
logger_syslog_level=2

ssid=RTWiFi-2015

# Operation mode (a = IEEE 802.11a, b = IEEE 802.11b, g = IEEE 802.11g,
# ad = IEEE 802.11ad (60 GHz); a/g options are used with IEEE 802.11n, too, to
# specify band)
# Note: RT-WiFi only supports 802.11 a/b/g
hw_mode=a
# Note: We recommend to run RT-WiFi in a less interference 5 GHz channels.
channel=157


macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=1

wpa=1
wpa_passphrase=your_password
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP

country_code=US

# Note: Do not enable 802.11n for RT-WiFi v 0.1
ieee80211n=0
```

Assuming that you put your hostapd configuration at */etc/hostapd/hostapd.conf*. Edit */ect/default/hostapd* and set the following line to automatically run hostpad.

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

isc-dhcp-server Configuration

Besides the authentication of 802.11, we need to set up a dhcp server which will dynamically allocate IP address for the connected stations. We assume the private IP of the AP's wireless network interface (wlan0) is 10.0.1.1, and the private network we use is in 10.0.1.0/24. Here's the configuration file of AP's wireless network interface.

```
#/etc/network/interfaces
auto wlan0
iface wlan0 inet static
address 10.0.1.1
netmask 255.255.255.0
```

Then we edit */etc/dhcp/dhcpd.conf* with the following configuration.

```
# Sample configuration file for ISC dhcpd for Debian

ddns-update-style none;

default-lease-time 600;
max-lease-time 7200;

log-facility local7;

subnet 10.0.1.0 netmask 255.255.255.0 {
      range 10.0.1.2 10.0.1.100;
      option domain-name-servers 128.83.130.204, 128.83.185.41;
      option routers 10.0.1.1;
}
```

Station side Configuration

In the station side, we first configure the wireless interface with the following setting. This configuration state that we'll use DHCP protocol to dynamically get IP address from a DHCP server, and the authentication process is proceeded by wpa_supplicant.

```
#/etc/network/interface
auto wlan0
iface wlan0 inet dhcp
wpa-conf /etc/wpa_supplicant.conf
```

Then we edit */etc/wpa_supplicant.conf* to configure wpa_supplication with the following settings.

```
#/etc/wpa_supplicant.conf
#Sample wpa-supplicant Configuration
network={
        ssid="RTWiFi-2015"
        scan_ssid=1
        key_mgmt=WPA-PSK
        psk="your_password"
}
```

After setting up both the AP and station side, you might want to test if the network is working by tools such as iperf.

Driver Build Process

After making sure the basic wireless setup is correct, we are ready to build and run RT-WiFi driver.
1. Get RT-WiFi driver source, and untar driver source code

```
$ tar -zxvf ./rt-wifi-v0.1.tgz ~/tmp/
```

2. Install tools for building Linux kernel

```
$ sudo apt-get install build-essential linux-headers-$(uname -r) libncurses5-dev libncursesw5-dev
```

- If your kernel version is newer than 3.13, please install older kernel, or current RT-WiFi driver may not work in your system.

```
$ sudo apt-get install linux-headers-3.13.0-53 linux-headers-3.13.0-53-generic \
  linux-image-3.13.0-53-generic linux-image-extra-3.13.0-53-generic
```

3. Set driver configuration

```
$ cd ~/tmp/RT-WiFi-v0.1
$ make defconfig-ath9k
$ make menuconfig
```

Enable the following options

- [Wireless LAN] -> [Atheros Wireless Cards] -> [RT-WiFi Support]
- [cfg80211 DebugFS entries]
- [Export mac80211 internals in DebugFS]
- save & exit

4. Configure RT-WiFi driver
Duplicate default driver configuration. To customize your RT-WiFi network, please refer the following section for RT-WiFi configuration.

```
$ cd ~/tmp/RT-WiFi-v0.1/drivers/net/wireless/ath/ath9k/
$ cp rt-wifi-local-sample.h rt-wifi-local.h
$ cp rt-wifi-sched-sample.h rt-wifi-sched.h
```

5. Build RT-WiFi driver

```
$ cd ~/tmp/RT-WiFi-v0.1/
$ make -jn (n is the # of cpus in your system)
$ sudo make install
$ sudo reboot
```

RT-WiFi Configuration

In this version of RT-WiFi driver, all the configurations are statically embedded in

the configuration header file. In order to make changes, please edit the following two files, recompile and reload the RT-WiFi module to your machines.
There are two configuration files, rt-wifi-sched.h and rt-wifi-local.h which are located at

```
./code/drivers/net/wireless/ath/ath9k/
```

RT-WiFi users may duplicate the sample file rt-wifi-sched-sample.h and rt-wifi-local-sample.h and make changes properly to create their desired TDMA behavior.

rt-wifi-sched.h

This configuration file controls the admissible stations and the TDMA schedule. The RT-WiFi AP only response the probe requests from the admissible RT-WiFI stations. In the RT-WiFi driver, **RT_WIFI_STAS[]** array is used to list the MAC address of the admissible RT-WiFi stations. In the following example, there are 3 RT-WiFi stations in this network.

```
/* Authorized stations */
static struct rt_wifi_sta RT_WIFI_STAS[] = {
        { .mac_addr = {0x00, 0x22, 0x5F, 0xF6, 0x11, 0x21} },  // station 0
        { .mac_addr = {0x00, 0x26, 0xB6, 0xF7, 0x72, 0x10} },  // station 1
        { .mac_addr = {0x00, 0x22, 0x5F, 0xF6, 0x0E, 0xA7} }  // station 2
};
```

In an RT-WiFi network, the transmission is ruled by a TDMA superframe. Superframe is a sequence of consecutive time slots. It defines the device's communication pattern with neighbors and will repeat itself infinitely. A superframe consists of multiple links, and each link describes the communication behavior within a time slot. There are four link types in RT-WiFi: transmit, receive, beacon and shared. A beacon link is reserved for RT-WiFi AP to broadcast beacon frame to all the stations in the network; and a shared link is for multiple nodes to compete for transmitting data frames. Currently, RT-WiFi only supports infrastructure mode. Therefore, a transmit link is used for a station to transmit data to the AP; a receive link of a station is used for receiving a data frame from the AP.

As shown in the following code snippet, *enum rt_wifi_link_type* defines 4 link types that we described in RT-WiFi network. *struct rt_wifi_sched* defines a link in a time slot. The *offset* in *struct rt_wifi_sched* describes the index of the time slot in the superframe. For a **RT_WIFI_TX** or **RT_WIFI_RX** link, *sta_id* describes the corresponding transmitting or receiving station. The *sta_id* is the index of a station in **RT_WIFI_STAS[]** array.

```
/* link scheduling */
enum rt_wifi_link_type{
```

```
        RT_WIFI_BEACON,
        RT_WIFI_TX,
        RT_WIFI_RX,
        RT_WIFI_SHARED,
};

struct rt_wifi_sched{
        enum rt_wifi_link_type type;
        u16 offset;
        u8 sta_id;
};
```

To configure TDMA superframe, edit **RT_WIFI_PRE_CONFIG_SCHED[]** array in rt-wifi-sched.h. The following example shows a superframe with 4 time slots. In time slot 0, station 0 transmits data frame to AP. In time slot 1, station 0 receives data from AP. Time slot 2 and 3 are reserved for beacon transmission.

```
static struct rt_wifi_sched RT_WIFI_PRE_CONFIG_SCHED[] = {
        { .type = RT_WIFI_RX, 0,  .sta_id = 0x00 },
        { .type = RT_WIFI_TX, 1,  .sta_id = 0x00 },
        { .type = RT_WIFI_BEACON, 2},
        { .type = RT_WIFI_BEACON, 3},
};
```

Please notice that in the default setup, hostapd will transmit a beacon frame for every 100 time unit. Please reserve **RT_WIFI_BEACON** slot accordingly to avoid collision with a beacon frame.

rt-wifi-local.h

This configuration file controls TDMA transmission parameters and debug information.

This table shows the meaning of the TDMA transmission parameters in rt-wifi-loca.h

Field	Note
RT_WIFI_TIME_SLOT_LEN	See the next table for length of TDMA time slot.
RT_WIFI_ENABLE_COEX	0: Disable co-existence mode. 1: Enable co-existence mode.
RT_WIFI_NUM_OF_TRIES	Number of transmission trials in a time slot.
RT_WIFI_ENABLE_ACK	0: A sender disregards an ACK frame for its transmission. 1: A sender waits for an ACK frame for its transmission.

This table shows the defined value for RT_WIFI_TIME_SLOT_LEN in rt-wifi-local.h Notice that 1 Time Unit = 1024 microseconds.

Value	Length of TDMA time slot
0	64 Time Unit.
1	32 Time Unit.
2	16 Time Unit.
3	8 Time Unit.
4	4 Time Unit.
5	2 Time Unit.
6	1 Time Unit.
7	512 microseconds.
8	256 microseconds.
9	128 microseconds.

Transmission rate

We assumes that the default rate control algorithm is Minstreal. Then we can use DebugFS to set transmission rate dynamically.

```
$ sudo echo desired_data_rate > /sys/kernel/debug/ieee80211/phy0/rc/fixed_rate_idx
```

Please refer to the following table for the value of **desired_data_rate**.

Transmission rate table for IEEE 802.11a and IEEE 802.11g.

Value	Data Rate
0	6Mbps
1	9Mbps
2	12Mbps
3	18Mbps
4	24Mbps
5	36Mbps
6	48Mbps
7	54Mbps

License

This driver is based on ath9k driver in backports driver v3.13.2 (https://www.kernel.org/pub/linux/kernel/projects/backports/stable/v3.13.2/). We follows the original license in ath9k driver for the modified files in ath9k driver. We claim the following license for the new developed RT-WiFi module.

Acknowledgement

We like to thank Dr. Tianji Li and contributors of ath9k-devel and freebsd-wireless mailing list for their fruitful discussion and valuable suggestions.

Please acknowledge the use of the code with a citation.

```
"RT-WiFi: Real-Time High-Speed Communication Protocol for Wireless Cyber-Physical Control Applications"
Yi-Hung Wei, Quan Leng, Song Han, Aloysius K. Mok, Wenlong Zhang, Masayoshi Tomizuka,
IEEE Real-Time Systems Symposium (RTSS), 2013.
```
