ESP8266 Sniffer应用设计说明

Version 0.3

Espressif Systems IOT Team

Table of Contents

1. Sniffer 模式介绍

ESP8266 可以进入混杂模式（sniffer），接收空中的 IEEE802.11 包。可支持如下 HT20 的包：

- 802.11b
- 802.11g
- 802.11n (MCS0 到 MCS7)
- AMPDU

以下类型不支持:

- HT40
- LDPC

尽管有些类型的 IEEE802.11 包是 ESP8266 不能完全接收的，但 ESP8266 可以获得它们的包长。

因此，sniffer 模式下，ESP8266 或者可以接收完整的包，或者可以获得包的长度:

- ESP8266 可完全接收的包，它包含：
  - 一定长度的 MAC 头信息 (包含了收发双方的 MAC 地址和加密方式)
  - 整个包的长度
- ESP8266 不可完全接收的包，它包含：
  - 整个包的长度

结构体 `RxControl` 和 `sniffer_buf` 分别用于表示了这两种类型的包。其中结构体 `sniffer_buf` 包含结构体 `RxControl`。

```
struct RxControl {
    signed rssi:8;              // signal intensity of packet
    unsigned rate:4;
    unsigned is_group:1;
    unsigned:1;
    unsigned sig_mode:2;        // 0:is 11n packet; 1:is not 11n packet;
    unsigned legacy_length:12; // if not 11n packet, shows length of packet.
    unsigned damatch0:1;
    unsigned damatch1:1;
    unsigned bssidmatch0:1;
    unsigned bssidmatch1:1;
    unsigned MCS:7;             // if is 11n packet, shows the modulation
                                // and code used (range from 0 to 76)
    unsigned CWB:1; // if is 11n packet, shows if is HT40 packet or not
    unsigned HT_length:16;// if is 11n packet, shows length of packet.
```

```
        unsigned Smoothing:1;
        unsigned Not_Sounding:1;
        unsigned:1;
        unsigned Aggregation:1;
        unsigned STBC:2;
        unsigned FEC_CODING:1; // if is 11n packet, shows if is LDPC packet or not.
        unsigned SGI:1;
        unsigned rxend_state:8;
        unsigned ampdu_cnt:8;
        unsigned channel:4; //which channel this packet in.
        unsigned:12;
    };

    struct LenSeq{
        u16 len; // length of packet
        u16 seq; // serial number of packet, the high 12bits are serial number,
                 //    low 14 bits are Fragment number (usually be 0)
        u8 addr3[6]; // the third address in packet
    };

    struct sniffer_buf{
        struct RxControl rx_ctrl;
        u8 buf[36]; // head of ieee80211 packet
        u16 cnt;     // number count of packet
        struct LenSeq lenseq[1];  //length of packet
    };

    struct sniffer_buf2{
        struct RxControl rx_ctrl;
        u8 buf[112];
        u16 cnt;
        u16 len;  //length of packet
    };
```

回调函数 `wifi_promiscuous_rx` 含两个参数 (`buf` 和 `len`)。`len` 表示 `buf` 的长度，分为三种情况：`len` = 128 ，`len` 为 10 的整数倍，`len` = 12:

LEN == 128 的情况

- `buf` 的数据是结构体 `sniffer_buf2`，该结构体对应的数据包是管理包，含有 112 字节的数据。

- `sniffer_buf2.cnt` 为 1。
- `sniffer_buf2.len` 为管理包的长度。

LEN 为 10 整数倍的情况

- `buf` 的数据是结构体 `sniffer_buf`，该结构体是比较可信的，它对应的数据包是通过 CRC 校验正确的。
- `sniffer_buf.cnt` 表示了该 `buf` 包含的包的个数，`len` 的值由 `sniffer_buf.cnt` 决定 。
  - `sniffer_buf.cnt==0`, 此 buf 无效；否则， `len = 50 + cnt * 10`
- `sniffer_buf.buf` 表示 IEEE802.11 包的前 36 字节。从成员 `sniffer_buf.lenseq[0]`开始，每一个 `lenseq` 结构体表示一个包长信息。
- 当 `sniffer_buf.cnt > 1`，由于该包是一个 AMPDU，认为每个 MPDU 的包头基本是相同的，因此没有给出所有的 MPDU 包头，只给出了每个包的长度 (从 MAC 包头开始到 FCS)。
- 该结构体中较为有用的信息有：包长、包的发送者和接收者、包头长度。

LEN == 12 的情况

- `buf` 的数据是一个结构体 `RxControl`，该结构体的是不太可信的，它无法表示包所属的发送和接收者，也无法判断该包的包头长度。
- 对于 AMPDU 包，也无法判断子包的个数和每个子包的长度。
- 该结构体中较为有用的信息有：包长， `rssi` 和 `FEC_CODING`.
- `RSSI` 和 `FEC_CODING` 可以用于评估是否是同一个设备所发。

总结

使用时要加快单个包的处理，否则，可能出现后续的一些包的丢失。

下图展示的是一个完整的 IEEE802.11 数据包的格式：

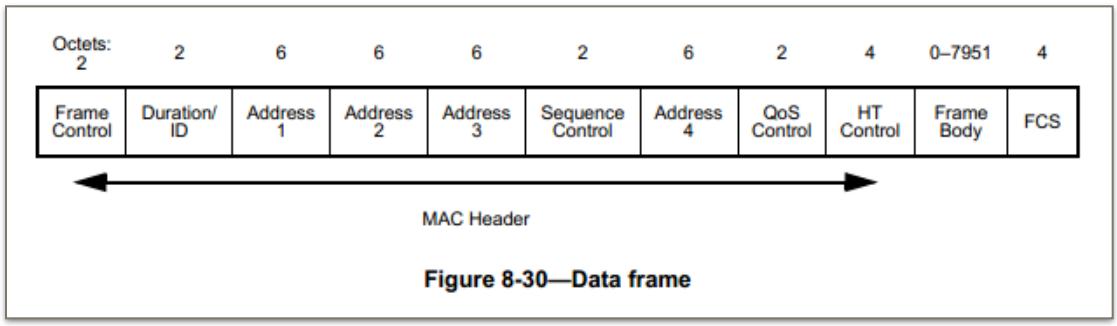

Figure 8-30—Data frame

- Data 帧的 MAC 包头的前 24 字节是必须有的:
  - `Address 4` 是否存在是由 `Frame Control` 中的 `FromDS` 和 `ToDS` 决定的;
  - `QoS Control` 是否存在是由 `Frame Control` 中的 `Subtype` 决定的;

  - `HT Control` 域是否存在是由 `Frame Control` 中的 `Order Field` 决定的;
  - 具体可参见 IEEE Std 80211-2012.
- 对于 WEP 加密的包，在 MAC 包头后面跟随 4 字节的 IV，在包的结尾 (FCS 前) 还有 4 字节的 ICV。
- 对于 TKIP 加密的包，在 MAC 包头后面跟随 4 字节的 IV 和 4 字节的 EIV，在包的结尾 (FCS 前) 还有 8 字节的 MIC 和 4 字节的 ICV。
- 对于 CCMP 加密的包，在 MAC 包头后面跟随8字节的 CCMP header，在包的结尾 (FCS 前) 还有 8 字节的 MIC。

2. Sniffer 的应用场景和相关问题

因为部分 AP 不转发 UDP 广播包到 WLAN 空口，所以只能监听手机发的 UDP 包。这些 UDP 包是手机发给 AP 并且已加密的。

应用场景1：IOT-device 能收到手机发出的所有的包。

手机和 AP 的连接工作在 11b、11g、11n HT20 模式时，并且手机到 AP 的距离大于手机到 IOT-device 的距离，则 IOT-device 能收到手机发出的所有的包。

IOT-device 的固件可以通过 MAC 地址和 MAC-header (及 MAC-crypt ion-header) 来过滤 UDP 包，也可过滤重传包。

同时，对 11n 的 AMPDU 包，也可得到每个子帧的长度和 MAC-header (及 MAC-crypt ion-header)

应用场景2：IOT-device 不能收到手机发出的所有的包。手机发包信号很强，但格式IOT-device不支持。

有两种情况：

Case 1：

手机到 AP 的距离远大于手机到 IOT-device 的距离。这时，手机发的高数据率的包 AP 能收到，但 IOT-device 收不到。

例如：手机发 MCS7 的包，AP 能正确接收，IOT-device 不能正确接收，但可以解出物理层包头（因为物理层包头是用低速率 6Mbps 编码）。

Case 2：

手机给 AP 发包为 IOT-device 不支持的格式：

- HT40；
- LDPC编码；
- 11n MCS8 以上即MIMO 2x2。

同样，IOT-device不能正确接收，但可以解出物理层包头HT-SIG。

在以上两种情况下，IOT-device 可收到 HT-SIG，其中包含物理层包长。利用此信息实现，需要注意以下几个问题：

- 当不使用 AMPDU 或 AMPDU 中只有一个子帧时，可以推测出 UDP 包长。如果手机端 APP 发送 UDP 包序列中的间隔较长 20mS~50mS，每个 UDP 包就会在不同的物理层包中，可能是只有一个子帧的 AMPDU 包。
- IOT-device 中的固件可以先用 RSSI 过滤其他设备发的包。
- 重传包需要用包序列中隐含的信息来过滤，即要保证连续两个包的长度都是不同的。例如可采用以下方案：
  - 每两个信息包之间用特定包分隔，称为分隔符包。
  - 奇数包长度 0~511，偶数包的长度大于 512~1023。

2. 手机 APP 设计

对**应用场景2**，手机 APP 要注意以下几点：

- 每个 UDP 包发送间隔为 20mS 或以上。
- 每两个信息包之间用特定包分隔，称为分隔符包。
- 每个信息包的信息有冗余，前后包可相互校验
- 序列开始时，有标志包。这样，APP 是不停地循环发送整个序列。
- AP 的 BSSID （即 MAC 地址）只需发送最低两个字节，IOT-device 可以扫描到。如果 AP 没有使用隐藏 SSID，SSID 也不用加在信息序列中。所以，要分析 AP beacon 来检查是否为 “隐藏SSID”
- UDP 包的长度要乘以 4。因为，有可能手机发送 AMPDU 中只包含一个子帧的情况。此时，包长会补齐为 4 的整数倍。

对**应用场景1**，手机APP可以尽快发包。所以，考虑手机APP在快速发送和间隔发送之间来回切换。手机APP不知道当前是手机WiFi发的包对 IOT-device 是应用场景1还是应用场景2。

3. IOT-device 上固件设计

对应用场景2，IOT-device 上固件设计要考虑：

(1) 用 RSSI 搜索 channel。先在最强的 channel 上搜索特征序列。

(2) 用 RSSI 过滤无用的包。要考虑空气中 10~15db的波动，主要是有些包会下降 10db 以上。开始时只收最强的包。找到特征序列头后，可以放宽波动范围。

(3) 可检查 HT-SIG 的 Aggregation bit，即为 AMPDU 包。

(4) AMPDU 中只能使用 CCMP (AES) 加密。

(5) 分隔符包长度要考虑不同 QoS、加密算法和 AMPDU 还会取 4 整数倍并加 4。

(6) 使用相对值获得信息，即用信息包长度减去分隔符包长度。这样就不用处理各种加密和 AMPDU 增加的长度。