How to use radiotap headers

Pointer to the radiotap include file

Radiotap headers are variable-length and extensible, you can get most of the information you need to know on them from:

```
./include/net/ieee80211_radiotap.h
```

This document gives an overview and warns on some corner cases.

Structure of the header

There is a fixed portion at the start which contains a u32 bitmap that defines if the possible argument associated with that bit is present or not. So if b0 of the it_present member of ieee80211_radiotap_header is set, it means that the header for argument index 0 (IEEE80211_RADIOTAP_TSFT) is present in the argument area.

```
< 8-byte ieee80211_radiotap_header >
[ <possible argument bitmap extensions ... > ]
[ <argument> ... ]
```

At the moment there are only 13 possible argument indexes defined, but in case we run out of space in the u32 it_present member, it is defined that b31 set indicates that there is another u32 bitmap following (shown as "possible argument bitmap extensions..." above), and the start of the arguments is moved forward 4 bytes each time.

Note also that the it_len member __le16 is set to the total number of bytes covered by the ieee80211_radiotap_header and any arguments following.

Requirements for arguments

After the fixed part of the header, the arguments follow for each argument index whose matching bit is set in the it_present member of ieee80211_radiotap_header.

- the arguments are all stored little-endian!
- the argument payload for a given argument index has a fixed size. So IEEE80211_RADIOTAP_TSFT being present always indicates an 8-byte argument is present. See the comments in ./include/net/ieee80211_radiotap.h for a nice breakdown of all the argument sizes
- the arguments must be aligned to a boundary of the argument size using padding. So a u16 argument must start on the next u16 boundary if it isn't already on one, a u32 must start on the next u32 boundary and so on.
- "alignment" is relative to the start of the ieee80211_radiotap_header, ie, the first byte of the radiotap header. The absolute alignment of that first byte isn't defined. So even if the whole radiotap header is starting at, eg, address 0x00000003, still the first byte of the radiotap header is treated as 0 for alignment purposes.
- the above point that there may be no absolute alignment for multibyte entities in the fixed radiotap header or the argument region means that you have to take special evasive action when trying to access these multibyte entities. Some arches like Blackfin cannot deal with an attempt to dereference, eg, a u16 pointer that is pointing to an odd address. Instead you have to use a kernel API get_unaligned() to dereference the pointer, which will do it bytewise on the arches that require that.
- The arguments for a given argument index can be a compound of multiple types together. For example IEEE80211_RADIOTAP_CHANNEL has an argument payload consisting of two u16s of total length 4. When this happens, the padding rule is applied dealing with a u16, NOT dealing with a 4-byte single entity.

Example valid radiotap header

```
0x00, 0x00, // <-- radiotap version + pad byte
0x0b, 0x00, // <- radiotap header length
0x04, 0x0c, 0x00, 0x00, // <-- bitmap
0x6c, // <-- rate (in 500kHz units)
0x0c, //<-- tx power
0x01 //<-- antenna
```

Using the Radiotap Parser

If you are having to parse a radiotap struct, you can radically simplify the job by using the radiotap parser that lives in net/wireless/radiotap.c and has its prototypes available in include/net/cfg80211.h. You use it like this:

```
#include <net/cfg80211.h>

/* buf points to the start of the radiotap header part */
```

```
int MyFunction(u8 * buf, int buflen)
{
        int pkt_rate_100kHz = 0, antenna = 0, pwr = 0;
        struct ieee80211_radiotap_iterator iterator;
        int ret = ieee80211_radiotap_iterator_init(&iterator, buf, buflen);

        while (!ret) {

                ret = ieee80211_radiotap_iterator_next(&iterator);

                if (ret)
                        continue;

                /* see if this argument is something we can use */

                switch (iterator.this_arg_index) {
                /*
                * You must take care when dereferencing iterator.this_arg
                * for multibyte types... the pointer is not aligned.  Use
                * get_unaligned((type *)iterator.this_arg) to dereference
                * iterator.this_arg for type "type" safely on all arches.
                */
                case IEEE80211_RADIOTAP_RATE:
                        /* radiotap "rate" u8 is in
                        * 500kbps units, eg, 0x02=1Mbps
                        */
                        pkt_rate_100kHz = (*iterator.this_arg) * 5;
                        break;

                case IEEE80211_RADIOTAP_ANTENNA:
                        /* radiotap uses 0 for 1st ant */
                        antenna = *iterator.this_arg);
                        break;

                case IEEE80211_RADIOTAP_DBM_TX_POWER:
                        pwr = *iterator.this_arg;
                        break;

                default:
                        break;
                }
        }  /* while more rt headers */

        if (ret != -ENOENT)
                return TXRX_DROP;

        /* discard the radiotap header part */
        buf += iterator.max_length;
        buflen -= iterator.max_length;

        ...

}
```

Andy Green <andy@warmcat.com>