

琴剑飘零

博客园 首页 新随笔 联系 订阅 管理

【Atheros】网卡驱动速率调整算法概述

我做网卡驱动，最主要的内容就是设计和改进速率调整算法，随着802.11协议簇的新标准越来越多，速率越来越高，调制编码方式也越来越多，一般来说，速率越高越可能丢包，速率越低越稳定，这是整体状况，但不是必然的规律，所以，只用固定的速率来发送显然是不合适的，这就需要速率调整算法来自己调节，信号比较好的时候，就用高速率来发送，信道状况不好了，就换用低速率来发，atheros驱动中提供了两种可选的速率调整算法，ath9k和minstrel，其中minstrel要好一些，后面我会分别根据源码解读minstrel和ath9k这两种算法，这一篇文章，只介绍一个重要的结构体，从而引出速率调整算法的任务。

接触过网络编程的朋友都不会对socket感到陌生，在数据链路层，有一个与之对应的结构体，叫做sk_buff，一般记作skb，skb中有一个域叫做control_buffer，也就是skb->cb，是一个48字节长度的内存区域，这个区域的设计，是为了协议栈中各个层存储一些私有的数据，数据包在不同层之间传递时，比如从网络层传输到链路层之后，这个域的数据就没有用了，可以放心地被链路层写入新的数据，链路层在把这个数据包交给物理层去发送的时候，需要指定一些参数，比如这个数据包要用20MHz还是40MHz去发送，用哪个速率去发送，如果发送失败了需要重传的话，最多重传多少次等等的信息，这些信息就存储在skb->cb中，存储的格式，是按照下面这个结构体来存的：



```
struct ieee80211_tx_info {
    /* common information */
    u32 flags;
    u8 band;

    u8 antenna_sel_tx;

    u16 ack_frame_id;

    union {
        struct {
            union {
                /* rate control */
                struct {
                    struct ieee80211_tx_rate rates[
                        IEEE80211_TX_MAX_RATES];
                    s8 rts_cts_rate_idx;
                };
                /* only needed before rate control */
                unsigned long jiffies;
            };
            /* NB: vif can be NULL for injected frames */
            struct ieee80211_vif *vif;
            struct ieee80211_key_conf *hw_key;
            struct ieee80211_sta *sta;
        } control;
        struct {
            struct ieee80211_tx_rate rates[IEEE80211_TX_MAX_RATES];
            u8 ampdu_ack_len;
            int ack_signal;
            u8 ampdu_len;
            /* 15 bytes free */
        } status;
        struct {
            struct ieee80211_tx_rate driver_rates[
                IEEE80211_TX_MAX_RATES];
            void *rate_driver_data[
```

公告

昵称：琴剑飘零
园龄：6年
粉丝：68
关注：12
+加关注

2017年9月						
<	日	一	二	三	四	五
	27	28	29	30	31	1
	3	4	5	6	7	8
	10	11	12	13	14	15
	17	18	19	20	21	22
	24	25	26	27	28	29
	1	2	3	4	5	6

搜索

找找看

谷歌搜索

常用链接

- 我的随笔
- 我的评论
- 我的参与
- 最新评论
- 我的标签

随笔分类

- Android(6)
- JavaSE/EE(11)
- Web技术(3)
- Web前端(6)
- 编程语言(2)
- 设备驱动(10)
- 设计模式(2)

随笔档案

- 2015年8月 (1)
- 2015年7月 (1)
- 2015年3月 (2)
- 2015年1月 (3)
- 2014年11月 (18)
- 2014年10月 (1)
- 2014年9月 (1)
- 2014年5月 (1)
- 2014年4月 (1)
- 2014年2月 (1)
- 2014年1月 (2)

```
IEEE80211_TX_INFO_RATE_DRIVER_DATA_SIZE / sizeof(void *));  
  
};  
  
void *driver_data[  
    IEEE80211_TX_INFO_DRIVER_DATA_SIZE / sizeof(void *));  
  
};  
};
```



其中，很多字段的意义不需要操心，只需要关注其中union的一部分（标红的union），这段内存区域可以通过control和status等方式访问，这个control和status就是我们关注的重点！

他们的核心又是同样的一个字段：我标红的ieee80211_tx_rate：

```
struct ieee80211_tx_rate {  
    s8 idx;  
    u8 count;  
    u8 flags;  
} __packed;
```

这个结构体代表一个速率，看字段名就很明显了：速率号、发送次数和一个标志位（标识带宽、SGI/LGI等），网卡可能会支持多速率重传，就是先用某个速率发送，如果几次都失败了就换第二个速率发，因此，底层在发包过程中，需要一个ieee80211_tx_rate的数组，速率调整算法的任务，就是把前面结构体中的这个字段填充好交给底层：

```
struct ieee80211_tx_rate rates[IEEE80211_TX_MAX_RATES];
```

回到前面那个结构体tx_info，它拥有一个union，主要包括control和status两个部分，control部分是在发送过程中被使用的，速率调整算法会：



```
struct ieee80211_tx_info *tx_info = IEEE80211_SKB_CB(skb);  
struct ieee80211_tx_rate *rates = tx_info->control.rates;  
  
rates[0].idx=0; rates[0].count=2; rates[0].flags=zzz0;  
rates[1].idx=1; rates[1].count=4; rates[1].flags=zzz1;  
rates[2].idx=2; rates[2].count=8; rates[2].flags=zzz2;
```



这样，我就指定了，发送时数据包依次使用0、1、2这三个速率最多发送2、4、8次，直到发送成功为止。那么发送完成之后，我们需要知道发送到底成功了没有，最后是哪个速率发送成功的等信息。所以这个结构体又会被底层返回给我们，此时，我们可以通过status访问之前设置的这些数据，只是每一项的count域被底层重写了，之前是表示每个速率的最大发送次数是多少，现在变成了每个速率的实际发送次数是多少。



```
struct ieee80211_tx_info *tx_info = IEEE80211_SKB_CB(skb);  
struct ieee80211_tx_rate *rates = tx_info->status.rates;  
  
// rates[0].count == 4  
// rates[1].count == 1  
// rates[2].count == 0
```



这就代表速率0发了4次，最终都失败了，速率1发了1次就成功了，当然速率2没有用到，发了0次。

所以速率调整算法的任务就是在发送过程中把tx_info->control.rates填充好，等发送结束后根据tx_info->status.rates来做速率的调整。

下面就依次介绍这两种算法：[Minstrel](#)、[Ath9k](#)。

分类: [设备驱动](#)

好文要顶

关注我

收藏该文



2013年11月 (2)

2012年10月 (2)

2012年2月 (1)

2011年10月 (1)

2011年9月 (3)

最新评论

1. Re: 【JavaEE】Springmvc+Spring+Hibernate整合及example

严重: Exception sending context initialize event to listener instance of class org.springframework.we.....

--qidiantianxia

2. Re: 【JavaEE】SSH+Spring Security+Spring oauth2整合及example

@奉先 搞定了吗？能发一个例子让看看吗？...

--星辰海

3. Re: 【python】获取高德地图省市区县列表

你好，请问你这种方法能爬到路名吗？

--vvvvvww

4. Re: 【JavaEE】SSH+Spring Security+Spring oauth2整合及example

搞定了,谢谢你的文章

--奉先

5. Re: 【JavaEE】SSH+Spring Security+Spring oauth2整合及example

你好,我集成了您的ssh简易版oauth,项目名是demo4ssh-security-oauth的项目,security的配置几乎一致,前几步都可以走的通,但在用code换取access-token的.....

--奉先

阅读排行榜


1. 【JavaEE】SSH+Spring Security+Spring oauth2整合及example(9146)
2. 【JavaEE】Springmvc+Spring整合及example(4143)
3. 【JavaEE】Springmvc+Spring+Hibernate整合及example(3618)
4. 【JWPlayer】官方JWPlayer去水印步骤(3607)
5. 【Android】开源项目UI控件分类汇总之ProgressBar(3565)

评论排行榜

1. 【Atheros】Ath9k速率调整算法源码走读(25)
2. 【JavaEE】SSH+Spring Security+Spring oauth2整合及example(21)
3. 【Atheros】如何在驱动中禁用ACK(14)
4. 【Atheros】minstrel速率调整算法源码走读(12)
5. 【设计模式】简单工厂-工厂方法-抽象工厂(11)

推荐排行榜

1. 【Android】百度地图自定义弹出窗口(5)
2. 【百度地图】显示从某站点出发的所有公交线路(5)
3. 【JavaEE】SSH+Spring Security+Spring oauth2整合及example(5)



琴剑飘零

关注 - 12

粉丝 - 68

+加关注

0

推荐

0

反对


4. 【JWPlayer】官方JWPlayer去水印步骤(3)
5. 【python】获取高德地图省市区县列表(2)

« 上一篇：[【Atheros】如何在驱动中禁用ACK](#)

» 下一篇：[【Atheros】minstrel速率调整算法源码走读](#)

posted @ 2014-11-12 21:00 琴剑飘零 阅读(956) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

 注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

- 【推荐】50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库
- 【推荐】腾讯云上实验室 1小时搭建人工智能应用
- 【推荐】可嵌入您系统的“在线Excel”！SpreadJS 纯前端表格控件
- 【推荐】阿里云“全民云计算”优惠升级



GPU云主机5折起

M60 原价1660/月 现价830/月 P40 原价3400/月 现价2100/月

[查看详情](#)

- 最新IT新闻：
- 谷歌新DOODLE纪念墨西哥国家歌舞团创始人阿玛莉亚
 - 乐视游戏或被冰穹互娱收购 2016年净利润仅2万元
 - Facebook AI研发主管：一味模仿人脑将阻碍AI的发展
 - 50位演唱嘉宾助阵！魅蓝6演唱会又回来了
 - 京东联合欧莱雅发布“包容美力计划” 助力残障群体就业
- » 更多新闻...



极光统计 多维洞察用户增长运营指标

- 最新知识库文章：
- Google 及其云智慧
 - 做到这一点，你也可以成为优秀的程序员
 - 写给立志做码农的大学生
 - 架构腐化之谜
 - 学会思考，而不只是编程
- » 更多知识库文章...