第 34 卷 第 1 期 计 算 机 工 程 2008 年 1 月
Vol.34 No.1 Computer Engineering January 2008
·网络与通信· 文章编号：1000—3428(2008)01—0140—02 文献标识码：A 中图分类号：TP393

Linux 内核 802.11 无线网络协议栈的设计与实现

朱 轶，赵 洁
（上海交通大学网络学院，上海 200030）

摘 要：对Linux(2.6 版)内核中IEEE 802.11 无线网络协议栈的设计和实现进行了介绍，包括无线网络协议栈在内核中的位置和主要功能、与底层硬件设备的协作和与用户配置工具的交互等。根据无线网卡硬件的特点，如延迟性大、可靠性差等阐明了该协议栈的设计要领和技术细节，并着重对其重要实现部分进行了数据结构的介绍和代码分析。

关键词：内核；网络协议栈；无线局域网；IEEE 802.11 标准

Design and Implementation of 802.11 Wireless Network Stack in Linux Kernel

Zhu Yi, Zhao Jie
(College of Network System, Shanghai Jiaotong University, Shanghai 200030)

【Abstract】 This paper introduces the design and implementation of the 802.11 network stack in the 2.6 Linux kernel including the position of the wireless network stack in the Linux kernel, the major functionality, and the communication with both the hardware and user space configuration tools. Since there is much difference between the wired and wireless hardwares, for example, high latency and low reliability, it also performs a code analysis for some important parts of the implementation.

【Key words】 kernel; network stack; WLAN; IEEE 802.11

近十几年来，Linux[1]以其开放源代码的特点伴随着互联网的飞速发展取得了前所未有的成功。2.6 版本的Linux内核更是在进程调度、内存管理和设备驱动等方面取得了功能和性能的长足进步。然而，在 2.6 Linux内核创立的最初，并没有对近年来日益普及的遵循IEEE802.11[2]标准的无线局域网进行很好的支持。其中最突出的问题就表现在内核中没有一个用于处理IEEE 802.11 报文的通用的无线网络协议栈。为了解决以上问题以便为Linux下无线网卡驱动程序的开发提供方便，笔者以及相关人员设计并实现了名为IEEE802.11 的无线网络协议栈(http://IEEE802.11.sf.net)。经过半年左右时间的调试和完善，Linus Torvalds最终将其加入到Linux 2.6.14 内核版本中。今天，IEEE802.11 已成为Linux内核中无线网络协议栈的标准实现。

1 IEEE802.11 协议栈的主要实现目标

作为内核中的通用网络协议栈，IEEE802.11 除了需要处理所有 IEEE 802.11 系列标准(802.11a/b/d/g/e/h/i/n)中所规定媒体访问子层(MAC)的部分，还必须支持大部分现有的无线网卡，并且保证在所有的体系结构上运行(需要考虑体系结构的字长度和字节序问题)。IEEE802.11 的实现目标如下：

(1)将 IEEE 802.11 协议中的类型和定义结构化为 C 语言的数据结构以方便使用。

(2)完成通用 802.11 数据报文的发送和接收处理。

(3)完成部分 802.11 控制报文的发送和接收处理。

(4)支持Linux内核通用 802.11 无线网络扩展[3]回调 函数。

(5)实现 802.11 协议中的国家和地区无线信道的检查和设置。

(6)完成 802.11 协议中加密和解密的软件实现[4]。

2 IEEE802.11 协议栈的适用范围

IEEE802.11 作为 Linux 内核标准的无线网络协议栈，可以支持工作在 BSS, IBSS 和 Monitor 模式(Master 模式，即无线网卡作为 AP 的功能尚不支持)下的所有无线网卡。考虑到各种无线网卡在设计中软硬件工作分配的不同，根据对 IEEE 802.11 MAC 功能实现的位置，可将现有的无线网卡分为 3 种: (1)FullMAC 将所有的功能实现放入硬件或固件种完成；(2)SoftMAC 将所有实现放入到主机中的驱动程序和操作系统中完成；(3)HalfMAC 介于二者之间，即一部分工作在硬件中实现而另一部分工作在软件中实现。IEEE802.11 协议栈在设计中考虑到了这个因素，它可以根据所使用的网卡设备在硬件上的特点而动态的选择和改变配置，从而达到了支持所有以上 3 种硬件设计的无线网卡的目的。

在实现的过程中，这种灵活的功能是通过回调函数来完成的。设备驱动程序在使用 IEEE802.11 协议栈时应设置 IEEE802.11 所需要的回调函数，例如.hard_start_xmit 方法即为网卡驱动程序的硬件发送函数。

IEEE802.11 协议栈还提供了一些控制开关用来为不同的设备驱动提供服务定制。例如，IPW2200 网卡支持硬件的 AES 加密和解密，其驱动就可以把对应的 IEEE802.11 数据结构中的.host_encrypt 和.host_decrypt 设置为空，这样 IEEE802.11 协议栈在处理发送和接收的过程中会省略掉相关的操作而交

作者简介：朱 轶(1977－)，男，硕士研究生，主研方向：计算机网络；赵 洁，学士

收稿日期：2007-02-15 **E-mail：**chuyee@gmail.com

由硬件完成。

3 IEEE802.11 网络协议栈的实现

3.1 IEEE802.11 设备的创建

普通的Linux下网络设备驱动程序需要调用alloc_netdev()或alloc_etherdev()来分配代表网络设备的struct net_device结构[5]。使用IEEE802.11协议栈的驱动程序需要调用alloc_IEEE802.11()。alloc_IEEE802.11()自身会调用alloc_etherdev()为设备分配相应的struct net_device结构，并且在net_device和设备私有数据结构(struct ipw_priv)空间中插入了用于本设备的IEEE802.11_device数据结构，如图1所示。

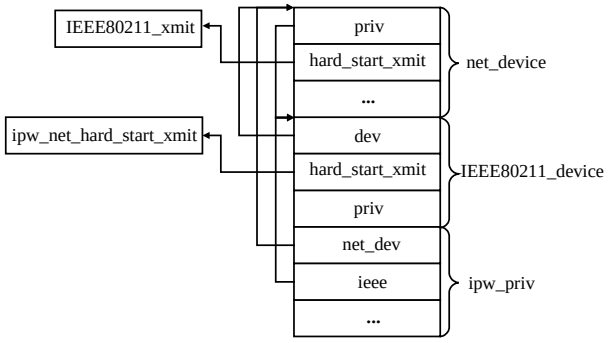

图1 IEEE802.11_priv 结构

除了对设备的IEEE802.11_device数据结构做初始化外，alloc_IEEE802.11()的另一个重要的工作是把设备所对应的net_device结构中的->*hard_start_xmit*指针指向了IEEE802.11协议栈的标准发送函数IEEE802.11_xmit()。这样对于内核来说，所有使用了IEEE802.11设备的硬件发送函数都指向了IEEE802.11协议栈标准的发送函数。当上层的网络协议栈(例如IP)或网络调度器(Qdisc)需要调用网卡设备的硬件发送接口时都会进入到这个入口点。IEEE802.11_xmit()在处理完IEEE 802.11协议的相关操作后会根据在IEEE802.11_device结构中的->*hard_start_xmit*指针把网络包传递给具体的设备硬件发送函数。

3.2 IEEE802.11 数据包的发送

IEEE802.11协议栈的发送函数是IEEE802.11_xmit()。802.11无线局域网(WLAN)与以太网有着很多相似之处，为了方便起见无线网卡设备在内核中的设备类型被定义为与以太网相同。由于IEEE802.11协议栈的发送函数是替换了标准网卡的硬件发送函数(net_device的.hard_start_xmit方法)而得以执行的(即在所有网络协议处理之后)，因此当IEEE802.11_xmit()被调度时待处理的网络包为以太网的格式。IEEE802.11_xmit()主要完成以下几部分工作：

(1)通过分析以太网的包头得出发送源和目的地的MAC地址及上层协议类型(如IP，ARP等)为数据包封装802.11 MAC协议头。

(2)为数据包封装SNAP协议头。与以太网协议(IEEE 802.3)不同，IEEE 802.11协议需要依赖于IEEE 802.2逻辑链路控制(LLC)封装来承载上层的协议。IEEE 802.2的子网获取协议(SNAP)用以完成此目的。

(3)判断是否需要设置RTS位。RTS和CTS是802.11的控制报文，可用于检查隐藏节点和802.11B、G网络间的通信保护。

(4)判断是否需要使用软件加密和软件数据包分片。若需要，根据加密模式选择调用加密子系统，先对未分片的整个数据包使用MSDU方式加密，再根据数据包分片限定大小对数据包进行软件分片并对每一个分片使用MPDU方式加密。值得注意的是在分片的情况下所有的分片都要有802.11 MAC协议头，但只有第一个分片上才有SNAP协议头。

(5)将处理过的802.11数据包提交给具体设备硬件驱动程序的发送接口。

3.3 IEEE802.11 数据包的接收

IEEE802.11协议栈的接收函数为IEEE802.11_rx()。使用了IEEE802.11协议栈的驱动程序应该在设备硬件接收处理程序(通常为中断处理后半部，即bottom half)的最后来调用此函数。这样，驱动程序只需要把硬件中的网络数据包复制到struct sk_buff形式的系统内存中，其余的工作都交给IEEE802.11_rx()来完成。这里IEEE802.11_rx()所处理的包格式为IEEE 802.11类型，它的主要工作包括：

(1)无线网络接收信号的统计。无线网络的信号强度，信噪比等信息的统计可以有效的反应当前无线网络的质量。网卡驱动程序负责将网络的信号信息随同网络数据包一同提交给IEEE802.11_rx()来进行统一的处理。

(2)IEEE 802.11协议头的帧控制(frame control)项定义了改数据包的类型，分片，加密等信息。协议栈通过分析这些内容来选择对数据包进行不同的操作。例如在WPA CCMP的网络里并把ipw2200设置成为软件解密，这时IEEE802.11_rx()所收到的数据包的帧控制项加密位标记(protected frame)为真，IEEE802.11_rx()就会对该数据包进行解密。在相同的情况下如果把ipw2200设置成硬件解密，由于ipw2200的网卡驱动程序在解密完成后会把对应的加密位标记清除，这样在IEEE802.11协议栈看来该数据包是为经过加密的，因此不会再进行重复解密。

(3)对数据包进行分片减少了包的长度，这样降低了802.11网络包与工作在相同频段(2.4GHz ISM)上的其他设备(如微波炉)的干扰(因为单个分片的碰撞概率减小)，从而达到了提高网络效率和吞吐量。IEEE802.11协议栈接收函数的一个重要功能是对那些分片过的数据包进行重组。IEEE 802.11协议对分片和重组提供了两方面的支持：1)在802.11协议头的帧控制项中有一个后续分片位(more frag)，该位标志为1的数据包表明这是一个分片并且在其后还有其他的分片。2)802.11协议头的序列控制项(Seq-ctl)由16个比特组成，前4个比特作为分片序号而后12个比特才为网络包序号。如果数据包是经过分片的，不同的分片拥有相同的网络包序号和不同的分片序号。在无分片的情况下，分片序号都为0。

在IEEE802.11的实现中，把已收到但还不能进行组装的分片放到一个缓冲中(ieee->frag_cache[])，在接收到所有的分片后对其进行重组并且清除掉该缓冲项。在收到每一个分片后协议栈都要对其做时间戳校验，如发现当前分片(第1个分片除外)与上一个分片接收的时间间隔过长(IEEE802.11定义为2秒钟)则清除该缓冲行。由于IEEE 802.11协议规定在大部分情况下(包括分片)接收端需主动确认(ACK)数据包，在这种情况下，发送端会选择重新发送这组分片。

(4)将处理过的网络包(skb)的类型设置成为802.2 LLC(见eth_type_trans()的实现)利用netif_rx()把网络包递交给逻辑链路层协议栈来处理。

3.4 IEEE802.11 协议栈的软件加密和解密

在加密算法上用到了已有的内核加密算法ARC4，AES和Michael MIC等(详见内核代码linux/crypto/)。IEEE802.11协议栈首先实现的是一个通用的加密算法的框架(crypto

(下转第144页)