802.11 网卡 Windows 驱动的设计与实现*

何 柳[1]，程 鹏[2]，陈 勇[1]，马洪亮[2]，吴 斌[2]
(1. 重庆邮电大学，重庆 400065；
2. 中国科学院微电子研究所，北京 100029)

摘 要：介绍了 802.11 系列协议的发展及异同，分析了 Windows 系统中的网络驱动模型，根据 NDIS 驱动模型设计并实现了 802.11 网卡 Windows 驱动程序，重点介绍了驱动中的数据收发队列的设计管理和协议状态的转化，并通过测试表明可以实现 802.11 协议的功能。

关键词：802.11；Windows；驱动程序
中图分类号：TN492 **文献标识码**：B **文章编号**：1674-7720(2013)04-0003-03

Design and accomplishment of Windows 802.11 network card driver

He Liu[1], Cheng Peng[2], Chen Yong[1], Ma Hongliang[2], Wu Bin[2]
(1.Chongqing Universty of Posts and Telecommunications, Chongqing 400065, China;
2. Institute of Microelectronics of Chinese Academy of Science, Beijing 100029, China)

Abstract: This paper introduces the difference among these 802.11 series protocols and analyzes the network driver model under Windows operation system. According to the principle of NDIS driver mode, it designs a 802.11 wireless net card driver program in Windows operation system with focuses on the design of managing send/receive packets and the design of state machine of 802.11 protocols. Finally the driver passes the function test of 802.11 protocols.

Key words: 802.11; Windows; driver program

近年来，无线上网逐渐成为了生活中不可或缺的部分。人们对无线网络的需求越来越强烈，而无线局域网(WLAN)技术的快速发展也适时地满足了人们的需求。无线局域网克服了有线网络存在的布线问题，但同时也导致网络越加复杂，需要研究和关注的内容越来越多。在无线局域网中，驱动程序的设计是一个很重要的环节。无线网卡驱动程序设计的优劣直接影响到整个无线局域网的传输速率和稳定性。文章介绍了 802.11 系列协议的发展和异同以及 Windows 操作系统下无线网卡驱动程序的设计模型，着重分析了驱动程序中收发队列的设计和管理，以及 802.11 协议中驱动层状态转化的设计。

1 802.11 协议分析

1997 年 11 月 26 日，IEEE 发布了第一个在国际上被认可的无线局域网协议——802.11 协议。随后又推出了 802.11b、802.11a、802.11g 等一系列物理层协议，目前最新的 802.11ac 协议正在逐渐完善中。表 1 为各个版本的 802.11 协议的简单比较。

表 1 中提到的速率为最高传输速率，每个协议又可提供多个速率值，以便适应不同的传输环境。例如 802.11b 协议提供了 4 种传输速率，分别为 11 Mb/s、5.5 Mb/s、2 Mb/s 和 1 Mb/s。在 802.11 系列协议中 802.11a 与 802.11b 因使用频段不同，相互之间无法进行通信[1-2]。

表 1 802.11 系列协议比较

协议名	频率/GHz	速率/(Mb/s)	推出时间/年
802.11	2.4	2	1997
802.11a	5	54	1999
802.11b	2.4	11	1999
802.11g	2.4	54	2003
802.11n	2.4/5	600	2009
802.11ac	5	1 300	2013

802.11 协议功能的实现需要物理层和 MAC 层协助完成。物理层以芯片的形式存在，主要完成无线信号的发送/接收功能。而 MAC 层主要以网络驱动程序和硬件协议加速器的形式存在，主要完成数据收发的管理、协议状态切换和维护，以及与操作系统的交互等功能，在实现时需要遵循操作系统所规定的网络驱动模型。

* 基金项目：无线移动通信国家重大专项(2010ZX03005-001-02)

2 Windows 网络驱动模型

在 Windows 系统中实现 802.11 协议时需要遵循 Windows NDIS(Network Driver Interface Specification)网络驱动程序接口规范。NDIS 规范分离了上层协议与底层接口,使得在设计无线网卡驱动时更加方便快捷。NDIS 规范将网络驱动程序划分为三个层次:协议驱动层、中间驱动层和小端口驱动层,其中中间驱动层根据实际情况可以不用实现。图 1 为其驱动框架。不同的驱动层间通过 NDIS 库进行通信,这样在设计各层驱动时不用考虑与其他层之间的交互细节,只需要遵循相应的接口即可[3]。

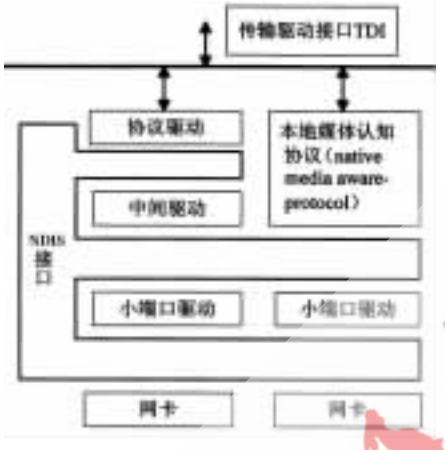

图 1 NDIS 驱动框架

按照 NDIS 规范,设计与实现 802.11 网卡驱动的主要工作在于编写小端口驱动。其中,只需要向 NDIS 注册指定的派遣函数,即可在 Windows 系统中增加 802.11 无线网络功能。一般而言,在小端口驱动中需要注册的派遣函数如表 2 所示[4]。

表 2 小端口驱动中需实现的派遣函数

派遣函数	功 能
MpCheckForHang()	检查是否挂起
MpHalt()	删除一个已注册的网卡设备
MpInitialize()	初始化
MPQueryInformation()	信息查询
MPReset()	网卡重置
MPReturnPacket()	处理上层驱动返回的包
MpMultipleSend()	发送数据
MPSetInformation()	向小端口设置信息
MPHandleInterrupt()	中断延时调用
MPIsr()	中断处理
MPCancelSendPackets()	取消发送
MPPnPEventNotify()	事件通知
MPShutdown()	恢复板卡到初始状态

在小端口驱动中,通过设置某个特定数据结构体中派遣函数指针的方式实现派遣函数的注册。在 WinXP 系统中,该数据结构为 NDIS_MINIPORT_CHARACTERISTICS,从 Vista 系统开始使用新的数据结构 PNDIS_MINIPORT_DRIVER_CHARACTERISTICS。下面以 WinXP 中的数据结构为例,介绍派遣函数的注册方式:

```
                                        //申明变量
NDIS_MINIPORT_CHARACTERISTICS NicChar;
                                        //变量内容置零
NdisZeroMemory(&NicChar, sizeof(NicChar));
                                        //注册初始化函数
NicChar.InitializeHandler = MpInitialize;
……;
                                        //注册发包函数
NicChar.SendPacketsHandler = MpMultipleSend;
```

当完成上述函数注册之后,上层执行网络相关操作时最终会调用到小端口层驱动的相应函数。如当上层查询网络信息时调用 MPQueryInformation()函数,而要发送数据包时则调用 MpMultipleSend()函数。在 802.11 网络驱动程序的设计中,数据收发队列的设计与管理是整个驱动程序设计开发的核心,其设计的优劣直接影响网络驱动程序的效率,下节针对该部分的设计展开论述。

3 数据收发队列的设计与管理

在无线网卡驱动的设计中,数据收发队列的管理方式和性能直接影响驱动程序的数据处理能力和工作性能。在设计数据收发队列时需要考虑下面几个内容[5]:内存分配和管理、收发队列的构造、队列资源的重用和同步。

内存分配和管理是无线网卡驱动程序设计中必需考虑的问题。数据收发队列中内存的分配有两种方式:(1)仅被驱动访问的内存,调用库函数 NdisAllocateMemory()进行非分页内存分配并返回内存的虚拟地址;(2)驱动和硬件都需进行访问的内存,调用库函数 NdisMAllocateSharedMemory()进行分配。此函数首先分配内存,然后将分配的内存进行物理映射,最后同时返回内存的虚拟地址和物理地址。

数据收发队列的设计和管理是无线网卡驱动中的难点。设计一个高效的数据收发队列需要在驱动和硬件 MAC 中进行交互设计。下面讲解此次设计的具体细节。

驱动层:

(1) 构建一个发送队列、一个空闲发送链表。空闲发送链表中包含所有未使用的发送资源,发送队列中包含所有准备发送的包。

(2) 在上层有数据包传入时,从发送链表中取出首节点。填充发送包的相关信息,插入发送队列尾部。

(3) 从发送队列头开始发送数据,直到发送队列为空。

(4) 等待发送完成中断,若产生则读取 HW_TX_MSDU 结构中硬件设置参数的值,根据状态值更新驱动状态。最后将节点插入空闲发送链表尾部,实现资源的重用。发送队列示意图如图 2 所示。

硬件层:

(1) 硬件访问 HW_TX_MSDU 结构,获取实际数据所

在的物理地址；然后根据硬件结构中的 next 指针判断是否有下一个需发送的包，若有则从 next 中获得下个包的物理地址；最后根据结构中相关参数设置对数据进行处理。

(2) 硬件处理完包，填充 HW_TX_MSDU 结构的相关状态变量。向系统发出一个中断，通知驱动包已处理完毕。

数据接收管理的设计与发送类似，这里不再赘述。实际收发链表以及队列的建立和管理与硬件的工作行为有十分密切的关系，在构造相关队列之前需了解硬件与主机交互数据的方式。因此与硬件工程师交流或熟读芯片数据手册是网卡驱动开发中的基础工作。无线网卡驱动程序的设计中除了收发包管理设计外，状态切换的设计也是一个需要重点注意的内容。

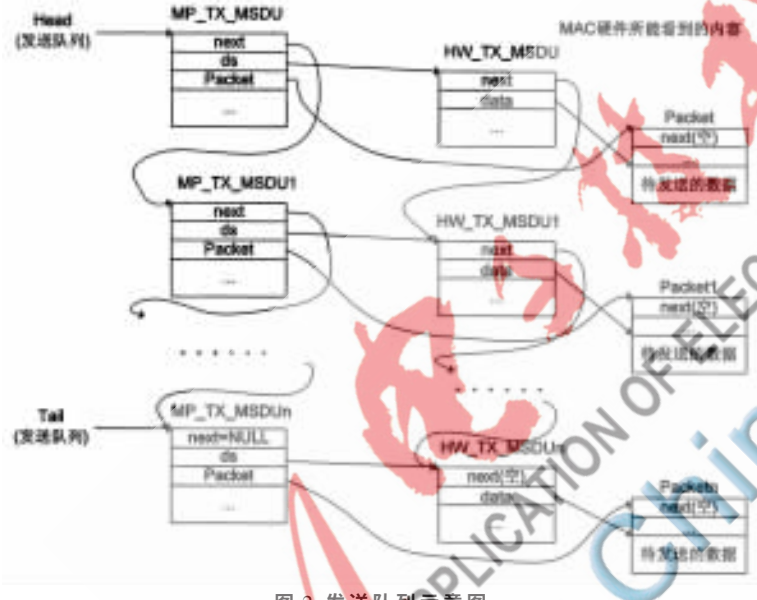

图 2 发送队列示意图

4 802.11 协议状态分析

802.11 协议中涉及到许多状态切换，状态切换的方式直接影响系统的稳定性。因此，设计一个合理的状态切换很有必要。在 802.11 协议的 MAC 层中有以下几种状态：初始态、加入(Connection)、认证、关联(Association)、运行和解关联[6]。

在无线网卡启动时首先启动扫描操作，通过扫描操作可以探测出覆盖范围内的所有基本服务集 (BSS)，然后根据用户的选择或者默认的设置连接/关联某个 BSS。由于无线链路的不稳定性，当前扫描到的 BSS 有可能在一段时间后消失，因此程序内部需要一个定时扫描信道的功能模块，用来实时更新当前可用的 BSS 列表。

在 802.11 协议状态设计时需首先分析出所有涉及到的状态，然后列出各个状态之间切换的方式，最后画出状态切换图表，用以指导具体代码的编写。图 3 为无线网卡驱动中 STA 模式下协议状态的切换。

图 3 无线网卡驱动状态机

本文提出的收发队列的设计和 802.11 协议中不同状态切换的设计对 Windows 下无线网卡驱动的设计有着一定的指导性作用。其中，收发包管理方式和状态切换设计已经在研发的芯片上进行了测试，证实了设计的有效性。

参考文献

[1] MATTHEW S G. 802.11 wireless networks the definitive guide[M]. O'Reilly, 2005.

[2] IEEE 802.11 protocol wireless LAN medium access control(MAC) and physical layer(PHY) Specifications[S]. IEEE, 2007.

[3] 贺鹏，李建东，陈彦辉. 带有 WDM 底层接口的 NDIS 微端口驱动程序实现方法的研究[J]. 现代电子技术，2004，27(2):93-95.

[4] Microsoft. Microsoft Windows driver kits[EB/OL]. [2009-12-xx]. http//www.micorsoft.com//wdk.

[5] 谭文，杨潇，邵坚磊. Windows 内核安全编程[M]. 北京：电子工业出版社，2009.

[6] 孙吉泉，鞠艳. 802.11MAC 层协议分析[J]. 中国科技信息，2009(14):134-135.

(收稿日期：2012-12-30)

作者简介：

何柳，男，1987 年生，硕士研究生，主要研究方向：Windows 内核驱动开发。

程鹏，男，1982 年生，助理研究员，主要研究方向：宽带无线通信系统。

陈勇，男，1963 年生，副教授，硕士生导师，主要研究方向：通信与信息系统。