Windows网络体系结构总结（转自看雪，作者：jbwang）

转载

标 题: 【原创】Windows网络体系结构总结
作 者: jbwang
时 间: 2009-11-23,11:22:25
链 接: http://bbs.pediy.com/showthread.php?t=101794

做了一些东西自己也看了一些书，最近总结了一下，想给大家分享一下，高手可以飞过了。如果有什么问题可以给小弟指正一下，多谢！

在介绍Windows网络体系架构之前，我首先介绍一下Windows中的两个重要编程规范——TDI，NDIS.，然后再介绍网络体系的架构。
TDI，Transport Driver Interface，传输驱动程序接口。\Windows\System32\Drivers\Tdi.sys
在实现网络API驱动程序时，由于牵涉到很多不同协议，会用到不同协议驱动提供的接口，使得开发的工作复杂化。所以Microsoft在网络API驱动程序和协议驱动之间又增加了一层TDI。TDI接口只是一种“将网络请求格式化成IRP，以及申请网络地址和数据通信”的做法规范化。遵从TDI标准的传输协议向他们的客户（如Socket emulator，Netbios emulator等）导出了TDI接口，有利于上下层之间的通信：
□ 一方面，对于TDI上层的网络API驱动程序就不需要使用所有协议驱动程序所提供的接口，直接使用TDI提供的统一接口。
□ 另一方面，对于下层协议驱动程序（也称为TDI Transport Provider传输器）直接由TDI接口来调用，发出请求。

在Windows VISTA版本之后，TDI就不再使用了，取而代之的是Windows filter platform和Winsock kernel。

NDIS，Network Driver Interface Specification，网络驱动程序接口规范，在操作系统中的位置
\Windows\System32\Drivers\NDIS.sys
当一个协议驱动程序想要按照其协议的格式解析网上读写的数据时，而这些数据必须通过网络适配器才能取得，期望协议驱动程序能够理解市场上的每一款网络适配器的细微区别是不可能的。所以在1989年，由Microsoft和3Com联合开发的了NDIS，使得协议驱动程序可以以一种与设备无关的方式来跟网络适配器驱动程序进行通信。遵从NDIS的网络适配器驱动程序称为NDIS minport driver。

NDIS规范实现了与TDI标准类似的功能，都是将复杂的下层调用规范化、标准化，大大提高了Windows操作系统的可扩展性和兼容性。也表现在两个方面：
□ 对于下层，让网络适配器制造商很easy的开发自己的设备驱动程序，也就是Ndis miniport driver。这些miniport driver直接利用NDIS提供的接口发送指令，NDIS对这些格式化的指令进行解析，做进一步处理。（这些处理就到了HAL了）
□ 对于上层，多个协议驱动程序与下层minport driver之间的通信，也都是通过统一的NDIS接口，NidsAllocatePacket，NdisSend等函数来收发数据。

废话两句：TDI和NDIS两大接口规范，有力的提升了Windows操作系统对不同设备厂商的支持，降低了设备厂商对设备驱动程序开发的难度；也增加了对于不同网络协议的支持，给用户更强大的网络功能支持。这种设计我们也可以在Windows存储管理中看到，从中我们似乎可以了解到一些，Windows操作系统在商业上取得成功的原因。Linux操作系统中没有这样的驱动层次结构。

设备制造商开发的Ndis miniport driver直接调用NDIS库中的接口函数，因此不需要考虑重入的问题，就是一个请求尚未结束的时候，新的请求又进来了。NDIS库对请求进行了序列化，但是这种序列化也妨碍了多处理器的扩展性。所以NDIS5中提供了非序列化的操作项。下面，我来介绍一下，Deserialized和Serialized minport driver的区别：
Deserialized NDIS miniport driver自己序列化对MinportXxx函数的操作，排队和管理多个并发请求的任务都由驱动程序自己来完成。而Serialized NDIS miniport driver以上的工作都是依赖于NDIS库来完成的。从性能角度
看，Deserialized NDIS miniport driver的性能是Serialized NDIS miniport driver性能的2倍多，所以到NDIS6.0之后的所有Miniport driver都是deserialize的。

以上是我参考MSDN以及自己的一些理解画出来的windows网络架构图，下面我就从上到下来简单介绍一下其中的各个层。

1. 网络应用程，Network applicantion，用户态的应用程序调用Windows操作系统提供的网络API，网络API包括：
a) Windows套接字（winsock）
b) 远程过程调用RPC
c) Web访问API
d) 命名管道和邮件槽
e) 其他网络API
这些API既可以在用户模式下实现，也可以同时在用户模式和内核模式下实现。从本质上说这些API是下层提供接口的另一层封装而已。
2. TDI Clients，传输驱动程序接口客户，是内核模式的设备驱动程序，用于实现网络API的内核部分。将网络API的请求转换成IRP，通过TDI标准格式化后，发送给下层的协议驱动（也就是TDI传输器）。从sockets emulator的架构图看到，TDI Clients的实现可以有用户态的部分，也有内核态的部分。AFD辅助功能驱动程序通过向协议驱动程序发送TDI IRP来执行网络套接字操作，比如发送和接受消息。AFD没有不是确定使用哪一个协议驱动，而是上层通知其要使用的协议名称，然后AFD去打开相应协议的设备对象。

3. TDI Transport Providers、TDI传输器、NDIS协议驱动程序、协议驱动程序，所有这些其实就是指的同一个东西，我在后面就称其为协议驱动程序。这个部分就是我们对某个协议的具体实现部分。做过网络协议开发的朋友一定知道，协议其实就是双发协商好的一套通信的规则。以IP协议为例，实际上就是对网络数据的一种处理方式，根据网络数据包的解析结构，做出相应的处理。Windows的tcpip.sys就实现了多个协议，ip、tcp、udp、arp、icmp、igmp，它为上层的TDI Clients提供了5个设备对象，用于访问使用这些协议，TDI Clients打开这些设备对象，向其发送IRP请求来实现自己的操作。通过DDK的DeviceTree我们可以得到这些设备对象
a) \Device\Rawip
b) \Device\Tcp
c) \Device\Udp
d) \Device\IPMULTICAST
e) \Device\Ip
协议驱动程序处理的数据是通过NDIS库中提供的接口来获取的，不需要发送IRP来取得。在DDK XP中提供了一个协议驱动程序了源程序Ndisuio，DDK XP后的版本提供的是Ndisport。在DriverEntry中我们可以看到，驱动程序一开始就注册了一个NDIS_PROTOCOL_CHARACTERISTICS，这个结构体中是一堆NdisXxxx函数。NDIS规范在这里就开始发挥它的作用了。
协议驱动程序的另一个作用就是监听网络数据，自己开发一个网络协议通过Ndis API获得所有的网络数据，但是不能够拦截网络数据，因为其他协议驱动也可以通过Nids API获取数据。一个典型的应用就是Winpcap了，使用NPF.SYS来捕获网络数据，并且做好充分缓冲处理，防止大数据量到来时出现数据包丢失的情况。详情情节winpcap的开源代码。
具体的协议驱动开发过程，我就不细述了，大家可以参看Ndisuio和DDK doc，我推荐boywhp的一篇文档《NDIS协议驱动开发》给大家。
4. NDIS，Network Driver Interface Specification，网络协议接口标准。从图中我们可以看到包裹在其中的两个驱动程序，一个是NDIS intermediate driver，NDIS中间层驱动程序，另一个是NDIS minport driver，小端口驱动程序。下面简单介绍一下这两个驱动程序：
a) Ndis intermediate driver，NDIS中间层驱动程序，对于上层的protocol driver它充当minport driver的作用，对于下层的minport driver它充当一个protocol driver的作用，所以在驱动程序DriverEntry中就注册NDIS_PROTOCOL_CHARACTERISTICS和NDIS_MINIPORT_CHARACTERISTICS，使用protocol characteristics中NDIS API从miniport driver那里取得数据包，再用miniport characteristics的NDIS API向上层的protocol driver发送数据包。
Nids intermediate driver最大的优势就是所有miniport driver的数据包都要通过它这里倒手给protocol driver，所以网络防火墙就看上了这块风水宝地。现在很多网络防火墙都使用NDIS intermediate driver做数据包的过滤和拦截工作，过滤的规则设置到MPSendPackets，PTReceive，PTReceiveRacket这三个函数。具体开发过程请大家参考DDK提供的PassThru源代码，www.ndis.com， 网上有很多相关的资料。
NDIS 6.0之后，filter driver就取代了Ndis intermediate driver，WDK中提供源码。
b) Ndis miniport driver一般是由设备厂商提供的，在DDK中也提供了miniport driver的一个例子e100bex，支持Intel EtherExpressTM PRO/100+ Ethernet PCI adapter 和Intel EtherExpressTM PRO/100B PCI adapter两款网络适配器。
5. 最后介绍一下总线，计算机总线有好几种，USB总线、ISA总线、PCI总线、虚拟总线等，一般都是以PCI总线作为根总线，在Windows系统中其他的总线可以理解为PCI总线上的一个设备。PCI总线作为根总线，其传输速度较高，可以达到133MB/S，显卡和网卡很多都是用PCI插槽。
PCI-ISA桥设备，也称为南桥，实现了ISA总线与PCI总线的桥接， 南桥还包括终端、IDE、USB、DMA等控制器设备。其中USB-HOST设备实现了USB总线和PCI总线的桥接。HOST/PCI桥称为北桥，是主处理器中心啊到基础PCI局部总线。南桥和北桥组成了主板的芯片组，通过芯片的扩展实现了多种总线与基础PCI局部总线的桥接。
总线驱动程序和PNP管理器实现了即插即用的功能，物理设备对象PDO就是由总线驱动程序产生的。