分类号：　　单位代码：10019

密　级：　　学　　号：1999020

中国农业大学

学位论文

PPP over Ethernet 协议及其在 Windows 系统上实现的研究

Research on PPPoE Protocol and Its Implementation on Windows Platforms

研　　究　　生：关　晓　华

指　导　教　师：陶　兰　教授

申请学位门类级别：工　学　硕士

专　业　名　称：计算机应用技术

研　究　方　向：计算机网络技术应用

所 在 院 （ 部 ）：电气信息学院

2002 年 3 月

摘要

本文简要地描述了 ADSL 互联网接入技术所采用的主要协议－PPPoE（Point-to-Point Protocol over Ethernet），然后提出了一种在 Windows2000 系统中实现 PPPoE 客户端软件的方案，并详细描述了其实现过程。

文章先对 ADSL 接入的各种方案做一比较，介绍了 PPPoE 模式的优势，然后结合国内国际研究现状以及市场需求，指出实现 PPPoE 客户端软件的必要性和紧迫性。

正文部分首先论述了 PPPoE 协议的主要内容，然后针对 Windows 的特性，提出一种解决方案，该方案利用现有 Windows 系统的资源，绕过 PPP 的细节而实现 PPPoE 协议并使 PPPoE 协议同 Windows 之间达到无缝结合；同时还解决了 PPPoE‘服务器/客户端’平台搭建问题，为软件的开发创造了良好的模拟环境。

接着描述了本课题的成果－PPPoE 客户端软件的总体结构、工作流程以及 PPPoE 协议驱动的各个功能部分。在对这些功能模块的介绍过程中还做了以下工作：

探讨了 PPPoE 协议在防范 DOS 攻击方面的支持；提出并实现了一种基于 PPP 协议的 PPPoE 连接控制方法；记载了虚拟拨号网卡完成安装和开始工作所要正确处理的 Windows 系统的 TAPI 序列；使 Windows 应用程序与核心驱动程序之间正确通讯。

本课题在国内较早对 PPPoE 协议进行了较为全面的研究，故对后继的开发和研究都有着较为重要的参考价值；该课题不仅实现了一个实用的 ADSL 工具，带来了可观的经济效益，更完善了在 PC 上实现的 PPPoE‘服务器/客户端’结构平台，从而为相关研发提供源代码级的支持。

关键字：ADSL , PPPoE, NDIS, PPP, 会话

Abstract

This paper briefly describes the PPPoE protocol (PPP over Ethernet，the dominant protocol adopted by ADSL service providers), presents strategies for implementing it under Windows and describes in detail an implementation of a PPPoE client.

The paper firstly introduces the advantages of PPPoE model by making a comparison among the existing solutions of Internet access through ADSL, and then points out the significance and urgency to implement PPPoE client software.

In the beginning of the main text, the paper discusses general contents of the PPPoE protocol, and then brings forward a solution to integrate the PPPoE protocol with Windows systems without touching PPP details. In the meanwhile, the paper also solve the problem of how to build the PPPoE C/S architecture as the necessary simulation environment for software developing.

It also describes the architecture of the PPPoE client software, its working process, and the function modules of the PPPoE protocol driver. The paper discusses following issues in the introduction of the modules:

DOS defense support in the PPPoE protocol; bring forwards and implement a PPPoE connection controlling method based on PPP; Windows TAPI-OIDs sequence which the virtual dialup adapter has to deal with properly; how to deal with the communication between kernel driver and the application

The paper makes an all-sided study on the PPPoE protocol for the first time in China, so it has relatively important values of reference for following study and developing; it not only realizes a practical tool for ADSL, which has tremendous economy profits, but also perfects the PPPoE C/S platform, which could support corresponding R&D actives at source code level.

Keywords：ADSL, PPPoE, NDIS, PPP, Session

目 录

第一章 概述

进入二十一世纪，随着社会信息化程度的提高，个人和企业用户对通信网络的要求也越来越高，网络应用的种类日益丰富，各种服务使得传统的窄带传输已不堪重负；由于网络传输速度过慢，WWW 甚至被称为‘ Wait Wait Wait ’。在此背景下，具有更高速率的数据传输技术——宽带技术的出现，使宽带建设的浪潮席卷全球。无论是各国政府还是电信运营商都把宽带网的推广作为发展的重点。

本章首先讨论目前宽带发展的现状，对主要的宽带接入技术作一介绍，重点对 ADSL 技术进行讨论，然后针对目前 ADSL 应用的现状引出本课题的内容和研究意义。

1.1 宽带网络的发展状况

目前，全球宽带接入显示出多元化的局面，其中较为成熟和活跃的接入技术有：ADSL、Cable Modem、ISDN、光纤接入、以太网、宽带无线接入等[1]。

1. ISDN（Integrated Services Digital Network 综合业务数字网）：

它是以综合数字电话网为基础发展而成的，能够提供端到端的数字连接。ISDN 将从一个用户终端到另一个用户终端之间的传输全部数字化，包括了用户部分，以数字形式统一处理各种业务，使用户可以获得数字化的优异性能。但是 ISDN 能提供的数据传输率最高为 128kbps，这还只是它的最佳工作频率，速率的提高同时也会引起收费的提高；而且构建 ISDN 网，需将现有的电话线改造成 ISDN 线路，工程浩大，这种方案在国外属于早期淘汰技术，在我国无论从技术上还是从市场上都属于过渡时期的解决方案。

2. ADSL（Asymmetric Digital Subscriber Line 非对称数字用户专线）[2]：

它的优点在于它利用现有的市内电话网和电话交换局的机房，可以降低施工和维护成本，并且对电话业务没有影响。它需要有一对 ADSL MODEM，其中一个被接到用户的计算机上，而另一台则安装在电话公司的通讯中心。根据所使用的 MODEM 型号的不同、以及用户端到通讯中心的距离长短， 在一对铜线上支持上行速率 640Kbps 到 1Mbps，下行速率 1Mbps 到 8Mbps，有效传输距离在 3－5 公里范围以内。也就是说 ADSL 方案的传输速度大约是 ISDN 方案的 50 倍，而且不需要改制线路；由于 Internet 上的应用大多是 C/S 型的，也就是说用户下载的信息往往比上载的信息（发送指令及控制信息）要多得多，所以非对称的速率匹配很适合用户上网高速冲浪或进行视频点播、远程局域网络等宽带应用。

世界各国的电信运营公司为了利用现有的用户线资源，正在积极开发宽带铜线接入技术。于是 ADSL 技术得以广泛地使用。

3. Cable Modem(电缆调制解调器)：

它是一种基于有线电视网络的技术，可在不影响有线电视广播的频带内实现对互联网信息的接入与访问。它的优点是带宽宽、速度快；

有线电视的同轴电缆是按单行道模式设计的，因此只允许信号从有线电视台传送到用户家中，不允许信号从用户家中反馈到有线电视台。而 Internet 联网却是一种双向的传输方式，有线

电视的解决方案需要花巨资改造基础线路，才能使它达到双向传输；用户在自己的电脑上也要接装高速电缆线 MODEM 才可上网浏览。

Cable Modem 实际上是一个粗糙的总线型网络，这意味着用户要和邻居共享有限的带宽，所以当同时上网人数增加时，上网速度就会变慢。局域网技术之所以淘汰总线型结构而采用星型结构，就是因为总线型网络固有的不可靠性。

4. 以太网接入技术：

以太网是目前应用最为广泛的局域网络传输方式，它采用基带传输，通过双绞线和传输设备，实现 10M／100M／1Gbps 的网络传输，是一种较为成熟的接入技术。

5. 宽带无线接入：

宽带无线接入技术具有带宽高和双向数据传输的特点，可提供多种宽带交互式数据及多媒体业务，而且其成本正在进一步降低。但目前无论在国际还是国内，宽带无线接入系统都属于刚刚起步的阶段。

宽带无线接入技术面临两大主要困难：一是基础建设费用庞大。通过若干个类蜂窝的服务区来提供业务，每个服务区至少需要建立一个基站，固定或移动的用户远端通过基站接入骨干网。由于需要建设大量基站，自然费用不低；二是用户数量比较少，运营成本高[3][4]。

6、光纤接入方式[5]：

与其他技术相比，光纤接入在带宽的接入速度和稳定性等方面都具有不可比拟的优势，但是由于成本太高，在目前很难得到普及。现阶段，运营商一般都是采用“光纤尽量接近用户”的方案，来逐步实施光纤接入。

目前全球宽带市场最为活跃的两种接入技术是利用有线电视网接入的 Cable Modem 和利用电话线改造的 ADSL。由于这两种技术只需将原来的线缆进行一定的改造而无需进行重新布线，所以相对于其它技术，它的前期投入较低，因此 ADSL 和 Cable Modem 深受投资者的欢迎。在宽带普及率最高的韩国，其宽带接入的主流技术也是 ADSL 和 Cable Modem。在日本，虽然 NTT(日本电报电话公共公司)一直在抑制 ADSL 的发展规模，但由于其它竞争者的强势进入，ADSL 在短期内就得到了飞速的发展。

对于消费者而言，ADSL 和 Cable Modem 的低廉价格和接入方式的简便，也使他们更乐意接受这两种接入方式。因此，无论是从市场的供给还是需求来看，近期宽带市场的竞争主要集中在 ADSL 和 Cable Modem 这两种接入模式[5]。

在中国，虽然广电拥有着广泛的有线电视用户，但由于广电还未实现台网分离，致使 Cable modem 接入方式在中国未能得到应有的发展。同时，由于我国并未实现电信与广电的双向准入，因此，Cable Modem 的发展还需要一个完善、有效的市场环境。

对中国电信而言，其 1.4 亿电话用户全都是通过铜双绞线接入网络的，这一庞大的硬件基础正是发挥 ADSL 优势的最佳平台，实施 ADSL 接入不需要对线路进行改造，就可以节省出大量的经费。每条 ADSL 线路的开通成本基本上是 ADSL 设备费用加上少量运营费用，而 Cable Modem 开通成本还要远远高于 ADSL 方案，从经济的角度上讲，ADSL 无疑是基于国内现有网络的一种现实的接入解决方案。

1.2 ADSL 技术简介

数字用户线（DSL，Digital Subscriber Line）技术是美国贝尔通信研究所于 1989 年为视频点播（VOD，Video on Demand）业务开发的利用双绞线传输高速数据的技术，由于 VOD 业务受挫而没有得到广泛的应用。近年来随着 Internet 的迅速发展，对固定连接的高速用户线需求日益高涨，基于双绞铜线的 xDSL 技术因其以低成本实现用户线高速化而重新崛起，打破了高速通信由

光纤独揽的局面[6]。

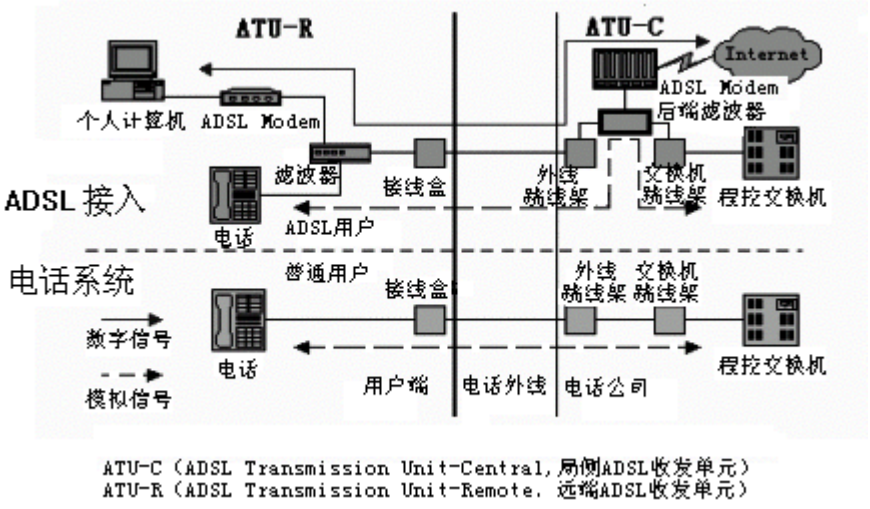
数字用户线有对称的和不对称的两大类应用，分别称为 SDSL（Symmetrical DSL）和 ADSL（Asymmetrical DSL），DSL 技术还包括 HDSL（high speed DSL）和 VDSL（very high speed DSL）。它们主要的区别体现在信号传输速度和距离的不同以及上行速率和下行速率对称性的不同这两个方面。 ADSL 因其在一对铜线上支持上行速率 640Kbps 到 1Mbps，下行速率 1Mbps 到 8Mbps，有效传输距离在 3～5 公里范围以内，非常适合 Internet 接入和 VOD，能够满足广大用户的需要，是最具有竞争力的一项技术。

ADSL 用其特有的调制解调硬件来连接现有双绞线连接的各端，它创建具有三个信道的管道。该管道具有一个高速下传信道(到用户端)，一个中速双工信道和一个 POTS（Plain Old Telephone Service，老式普通电话业务）信道，POTS 信道(4KHz)用以保证即使 ADSL 连接失败了，语音通信仍能正常运转。高速和中速信道均可以复用以创建多个低速通道。这三个信道以分频技术共用传统电话线从而实现高速传输。

在铜线上传输数据，衰减和串音是决定 ADSL 性能的两项标准损伤。传输速率越高，它们对信号的影响也越大，因此 ADSL 的有效传输距离随着传输速率的提高而缩短，这是它的主要缺点[7]。

ADSL 接入网线路长度若为 5.5km，则可覆盖 80%以上的现有电话用户 线路长度若为 3.7km，则可覆盖 50%以上的现有用户，用户小区以外的分散用户可通过基于光纤的集线器节点接入到网络中[8] [9]。

图表 1 是 ADSL 接入的一个具体实例，以及跟传统电话系统的比较。



图表 1 ADSL 的实物连接示意图

从该图可以看出，数字信号在电话局并不象普通拨号那样通过程控交换机，这减轻了交换机的负载；物理连接是永久的，一直在线，意味着使用 ADSL 上网并不需要缴付另外的电话费；也不会出现拨号占线的现象；用户在上网的同时还可以打电话，即使断电后也可以正常打电话，这又同 ISDN 不同；而且用户独享带宽，而不像 Cable Modem 那样得在总线网上与邻居共享带宽[3]。

1.3 ADSL 的应用方案及 PPPoE 的提出

PC 与 ADSL Modem 配合接入高速网络的具体实现也存在着多种方案[10][11]。

1.桥接接入方式（RFC1483）[12]

最初，RFC1483 标准的制定是为了实现网络层上多种协议的数据包在 ATM 网络上的封装传送。它已被广泛使用于 ATM 技术中，现已成为在 ATM 网络上处理上层多种协议数据包的封装标准。ADSL 接入依托于 ATM 骨干网络，在接入侧上继承了许多 ATM 技术的特点和优点，所以 ATM 网络上承载数据包的各种标准很自然地被 ADSL 接入技术所采用，使得 ADSL 的 RFC1483 桥接接入方式目前已成为 ADSL 宽带接入的最基本形式。在协议模型上，在数据链路层对网络层的数据包进行 LLC/SNAP 的封装，以此来指明上层所应用的协议类型，因此可以适用于网络层上的多协议传送。它仿真了以太网的桥接功能，在形式上这种接入方式相当于将用户侧的终端设备直接挂接在网络侧的网桥设备上。

处于数据链路层的桥接器在工作原理上仍然采用广播学习的方式，通过收集、记录各用户网卡的 MAC 地址信息进行通信。因此，RFC1483－Bridged 的接入可以实现远端整个以太网的共同接入，且组网经济简便。用户侧最大用户接入数的限制取决于 ADSL Modem 内部桥接记录存储器的实际容量。当然用户实际接入数还要考虑到 IP 子网的划分范围，这可以满足应用的实际需要。但是，应该看到，采用 RFC1483－Bridged 可以实现多用户的接入，但无法实现动态的 IP 地址分配，IP 地址资源利用率很低。同时，类似于实际的局域网，RFC1483 桥接接入不可避免地会引入大量的广播信息，引起整个网络效率的下降，各用户所获得的实际带宽抖动很大，很难完成对服务质量保证有严格要求的业务。

2.经典 IP 接入方式 （RFC1577）[13]

类似于 RFC1483 标准，RFC1577 也是在 ATM 网络上承载 IP 协议的标准规范。在协议中，它明确了该标准仅仅针对于网络层的 IP 协议，用 IP 路由转发实现相互之间的通信，因此也被称之为 IP over ATM。IP 包在数据链路层的封装处理上，RFC1577 仍然利用 RFC1483 LLC/SNAP 的数据封装方式对 IP 包进行封装处理，也被人们称之为 RFC1483－Router 接入方式。要实现 RFC1577 的 ADSL 宽带接入，用户侧的终端通常采用价格比较昂贵的 ATM25 网卡，同时该网卡供应商还必须提供让该网卡支持 RFC1577 协议的驱动程序。这在很大程度上限制了 RFC1577 在 ADSL 接入方面的推广使用。RFC1577 接入方式是由终端上的 ATM25 网卡完成对上层 IP 包的整个封装处理过程，最终形成 ATM 信元流传送至 ADSL Modem。因此，在这种接入方式中 ADSL Modem 仅仅作为 ATM 传输的一个端点。同时由于 ADSL Modem 内部没有 ATM 交叉连接的功能，如不使用附加的 ATM 设备将无法实现终端多用户的同时接入。但是，可以利用终端 ATM25 网卡的 ATM 特性来实现一些以太网卡无法实现的功能，比如：利用 ATM25 网卡以上行 RFC1577 接入请求，下行纯 ATM 信元流的形式来实现宽带视频点播应用，使 VOD 视频点播质量得到大大的提高。

3.PPP over ATM 接入方式 （RFC2364）[14]

以上讨论了 RFC1483 和 RFC1577 两种利用 ATM 早期标准完成静态 IP 接入的技术。对于一些有固定 IP 需求的专线用户，采用以上两种接入方式能够很好的实现。但是对于众多的普通接入用户而言，仍然利用静态 IP 方式实现宽带接入对于宽带接入运营商而言是很难接受的，尤其是在目前公网 IP 地址紧缺的情况下。因此，人们很自然地想利用窄带拨号动态分配 IP 地址的 PPP 接入技术应用到宽带的 ADSL 接入中来。因此 IETF（Internet Engineering Task Force，Internet 工程任务组）制定了利用 PPP 技术完成宽带拨号接入的标准规范——RFC2364，也称为 PPP over ATM。同时，在接入技术的发展推动下，网络侧的高性能宽带接入服务器利用 RFC2364 宽带接入技术能够很好地实现诸如业务选择、VPN 利用等一系列附带的特殊功能，为宽带接入的发展提供了更广阔的前景。

采用 PPPoA 的接入技术，由 PC 终端直接发起 PPP 呼叫，用户侧 ATM25 网卡在收到上层的 PPP 包后，根据 RFC2364 封装标准对 PPP 包进行 AAL5 层封装处理形成 ATM 信元流。ATM 信元透过 ADSL Modem 传送到网络侧的宽带接入服务器上，完成授权、认证、分配 IP 地址和计费等一系列 PPP 接入过程。从实现上看，ADSL Modem 也是仅仅作为 ATM 信元传送的一个端点。同时，要实现 PPPoA 的接入，用户侧仍然要求使用 ATM25 网卡，这就要求网卡供应商也必须提供相应的专用 PPPoA 驱动程序。类似 RFC1577 接入方式，在实现多用户同时接入方面，PPPoA 的接入方式也因 ATM25 网卡的自身局限性而无法实现。因此，PPPoA 虽然成功地解决了诸如动态 IP 地址分配和计费方面的一系列宽带接入问题，但是由于用户终端仍需要额外的网络设备和相应的驱动程序，事实上 PPPoA 这种宽带接入形式并没有得到大规模的推广应用。

4.点对点隧道技术转 PPPoA 接入方式 （PPTP－PPP）

最初 PPTP（Point-to-Point Tunneling Protocol，点对点的隧道协议）技术的出现是为了出于网络安全和 VPN 选择方面的考虑，是对 PPP 技术的一种扩展。为了充分利用现有的以太网络资源，结合 PPPoA 接入技术的优势，一些 ADSL 厂商将 PPTP 的技术引入到宽带接入应用中。它在 ADSL 用户侧利用 PPTP 技术（后来 IETF 将 PPTP 与 L2F 合并制定了 L2TP 标准规范）通过内部的以太网络在 ADSL Modem 与用户 PC 之间建立 IP 隧道传送用户终端发出的 PPP 请求。在 ADSL Modem 中终结 IP 隧道，提取 PC 终端发出的 PPP 包并相应地利用 RFC2364 标准进行封装处理，传送至远端宽带接入服务器完成基于 PPP 技术的授权、认证、计费和动态的 IP 地址分配等一系列过程。可见，PPTP－PPP 的接入在网络侧的实现与 PPPoA 是完全一致的。通过用户侧的 IP 隧道技术不仅可以有效地利用了现有的局域网络资源实现多用户的同时接入，而且在 ADSL Modem 中通过一些特殊的设置还可以完成简单的 VPN 选择功能。在驱动程序方面，用户 PC 终端可以利用现有 WIN98/NT 操作系统自带的虚拟专用网络适配器实现 PPTP 接入，无需再另行购买，具有很强的实用性。

这种接入方式的协议栈过于复杂，从而影响接入的实际性能。从功能上看，ADSL Modem 内部不仅要终结 PPTP 的 IP 隧道，而且要向网络侧发起 PPP 的连接。而作为 ADSL 终端设备的 ADSL Modem 在包处理能力上毕竟有限，这将大大降低用户接入的实际速率，几乎无法实现 VOD 视频点播这类有高带宽、高质量要求的业务。

其次，PPTP－PPP 的接入技术在支持用户数方面也限制，比如：阿尔卡特 ADSL Modem 最多支持 14 个 PPTP－PPP 的数据流。最后，从实际的组网角度上看，要完成每一终端用户的 PPP 接入都必须在 ADSL Modem 和 ATM 网络中创建传送该 PPP 包的相应 PVC，网络实现过于复杂。

5.PPPoE 接入方式 （RFC2516）[15]

针对以上几种接入方式存在的困难，Red Back 、UUNET、Routeware 等公司提出了一种新技术 PPPoE（PPP over Ethernet），IETF 又于 1999 年 2 月进一步制定了 PPPoE 的技术规范。

在实际应用上，PPPoE 利用以太网络的工作机理，将 ADSL Modem 的 10BASE－T 接口与内部以太网络互联，在 ADSL Modem 中采用 RFC1483 的桥接封装方式对终端发出的 PPP 包进行 LLC/SNAP 封装后，通过连结两端的 PVC 在 ADSL Modem 与网络侧的宽带接入服务器之间建立连接，实现 PPP 的动态接入。PPPoE 接入利用在网络侧和 ADSL Modem 之间的一条 PVC 就可以完成以太网络上多用户的共同接入，实用方便，实际组网方式也很简单，大大降低了网络的复杂程度。它总结了以往 PPP 宽带接入方式的特点和要求，兼顾了对用户终端的硬件要求，同时克服了在 PPTP－PPP 传输性能上的不足，提高了 ADSL 宽带接入的总体性能。因此，PPPoE 技术规范的完成得到了广泛的支持，目前成为宽带接入运营商首选的宽带接入方式[16]。

图表 2 对以上各种 ADSL 接入技术进行了综合比较分析，通过以下的比较结果，不难看出 PPPoE 模式的优越性。

因素 / 方案	IP 的获得	用户端所需的支持	是否支持多用户？	多用户下 PVC 的个数	驱动程序	传输性能	典型应用
RFC1483	静态	以太网接口	是	1	不需要	理想	远程以太网的静态接入
RFC1577	静态	ATM 接口	否	0	由 ATM 网卡商提供	很好	视频点播
PPPoA	动态	ATM 接口	否	0	由 ATM 网卡商提供	理想	ATM 卡远程拨号接入
PPPTP-PPPoA	动态	以太网接口	是	1/用户	需要，Windows 系统自带	一般	现有局域网用户的拨号接入
PPPoE	动态	以太网接口	是	1	需要，由第三方提供	理想	以太网用户的拨号接入

图表 2 各种 ADSL 接入模式的性能比较

1.4 ADSL 和 PPPoE 的现状以及待解决的问题

1.4.1 ADSL 的应用现状

ADSL 技术可以提供的服务包括高速互联网接入、视频点播、网上游戏、远程医疗、远程教学和远程可视会议等。

在英国，最大的卫星广播公司 B Sky B 与英国电信合作，在英国家庭中大力推广 ADSL 业务，利用 ADSL 线路向用户提供多路电视广播和 Internet 的接入业务。英国广播(BT)公司还投入 7 亿英镑资金，计划到 2002 年让英国全国 70%的家庭接通 ADSL 线路。

在日本，截至 2001 年 5 月末，ADSL 用户已达到 18 万户，Yahoo 日本公司自 2001 年 6 月开展 ADSL 业务以来，至 2001 年 10 月 5 日，其 ADSL 预约用户已达 111.8 万户。

韩国是目前宽带发展最快的国家，也是 ADSL 应用最迅速的国家。根据国际电联的统计，ADSL 在韩国所占比例已达 25%，用户高达 500 万户。

美国的现有家庭 ADSL 用户已超过 500 万户；加拿大 ADSL 用户数亦已达 400 万户。

据中国互联网络信息中心在 2002 年 1 月发表的《第九次中国互联网络统计调查报告》，截至到 2001 年 12 月 31 日，我国上网用户总人数已达约 3370 万人，其中专线上网的用户人数为 672 万，拨号上网的用户人数为 2133 万，同时使用专线与拨号的用户人数为 565 万。而仅仅在半年之前（2001 年 7 月），这些数字还分别是 1690 万，258 万，1176 万和 256 万[17]。

在我国，据蓝田市场研究公司最近进行的全国信息产业大型调查结果显示，在单位宽带接入市场中，有 68.7%的单位用户采取了 ADSL 接入方式。据赛迪专题预测，2002 年网民有 30%左

右使用 ADSL（包括原用 Modem 的网民），也就是说大约将有 900 多万的 ADSL 的潜在市场，市场潜力十分乐观[17]。

在这样的应用前景下，困扰着 ADSL 服务商的却是用户数量相对增长过缓，市场开发不快。这是由多种因素造成的，其中价格因素占了很大比重，据中国互联网络统计调查报告，当前互联网有 50%以上用户的月收入在千元以下，73.7%的用户是全自费，而在我国 ADSL 一次性初装费都超过了 1000 元。如此高的服务费用既有硬件的原因，也有用户数目太少而造成平均服务成本较高的原因。

1.4.2 PPPoE 协议的现状

目前几乎所有的 ADSL 服务提供商都是利用 PPPoE 协议向用户提供服务。

进行 PPPoE 的测试和分析及 PPPoE 协议的升级，需要主机/客户机相配合，而服务端设备是很昂贵的，故对它的研究与发展主要是由 Red Back 、UUNET、Routeware 这几家从事网络硬件设备的公司合作或独立进行，目前已推出嵌入支持 PPPoE 的路由器和服务集中器[11]，而 PPPoE 协议本身的版本还是 1.0。至于其他组织及个人所作有关 PPPoE 的研究讨论则主要集中在产品层面上。

因为国外的 ADSL 发展的较早，市场上早就有了 PPPoE Client 端产品。目前最常用基于 Windows 操作系统的 PPPoE 软件有以下几个，它们都完全支持 Windows98，Windows NT 和 Windows 2000。

EnterNet300：由 Efficient Networks 开发，目前最新版本为 1.41。它具有独立的 PPP 协议，可以不依赖操作系统（如 Windows）中的拨号网络，直接驱动网卡连接 ISP，因此它是目前最通用和流行的 PPPoE 软件。多家特大型 ISP 如法国电信，中国电信都采用它。网址：http://www.nts.com/

WinPoET：由 PPPoE 协议起草者之一 RouterWare 公司开发，目前最新版本为 2.0。它需要通过操作系统自身的 PPP 拨号协议来支持完成 PPPoE 的连接，也是许多 ISP 使用的 PPPoE 软件。网址：http://www.windriver.com/ivasion/ 。

RasPPPoE：由德国人 Robert Schlabbach 个人开发的免费软件，目前最新版本为 0.96。它小巧精干，没有自己的界面和连接程序，主要部分是一个协议驱动程序，依靠同标准的拨号网络合作来连接 ISP，它在使用上和普通 MODEM 一样简单。另一个特点在于它提供模拟服务器的功能，即经过适当配置可以接受来自客户端的 PPPoE 拨号接入，这同其它 PPPoE 客户端软件的最大区别。网址：http://user.cs.tu-berlin.de/~normanb/。

另外在 Redhat Linux v7.0 已经内嵌 Roaring Penguin 公司的 RP-PPPoE[18]作为 PPPoE 的客户端软件。

在我国，硬件设备如路由器和 ADSL Modem 已有深圳华为等多家生成商提供，而在 Windows Client 端的软件都是使用国外的 Enternet 及其 OEM 版本。

当前的 ADSL 设备提供厂商每提供一台设备，都必须提供相应的驱动程序，而这样的驱动每份拷贝都需向 PPPoE 驱动软件开发商支付 3-10 美元的分发费，这一费用 ISP 提供商将全部计入硬件投资中，最终的承担者仍然是用户。

在我国主要的 ADSL 用户都是家庭用户，操作系统大部分是 Windows，国外非共享的 PPPoE 软件自然价格较高，而共享软件（RASPPPoE）则不提供技术支持，出现问题又无处可询。在这种情况下，急切地需要一款低价位，有完备技术支持的 PPPoE 客户端软件，唯一的解决方案就是自主开发一套 PPPoE 客户端软件。

1.5 选题的意义及工作内容

1.5.1 选题的意义

本文以“ PPP over Ethernet 协议及其在 Windows 系统上的实现的研究 ”为题进行的研究具有如下的现实意义和实用价值：

1.追踪国际先进技术。从全球角度出发观察自己所处的位置，选择最有利的切入点，才有机会赶超世界技术潮流。从通讯业来讲，ADSL 正处于蓬勃发展阶段，PPPoE 协议的制定也只有一年多的时间，很多方面需要完善。对 PPPoE 协议的研究既可以切入各种网络协议的深层，又同软件开发有效结合起来，通过研究探索出一些 Windows 尚未公开的技术内幕，还可以做出实用的工具，为以后在 ADSL 领域的研究提供依据。另外，PPPoE 不拘泥于 ADSL 应用，由于它的设备无关性，可以将其推广到 Cable Modem 或其他需要计时计费的网络接入模式，这都需要切实地实现该协议，才能得到第一手的资料，从而在新产品的开发层面上赶超世界水平。

2.技术本土化。加入 WTO 后，本国和国际之间技术力量的竞争会空前加强，技术本土化也可以视为新时代的‘ 拿来主义 ’，只有走‘ 学习－消化－应用－发展 ’的路线，我们才能在激烈的竞争中存活并发展。对任何先进技术跟踪了解的最终目的都是为此项技术的实用化。

对我国目前的现状而言，ADSL 的配套软件基本上是美国 NTS 公司的 Enternet 的 OEM 产品。开发出自己的产品，既可以建立 PPPoE C/S 平台，又可以基于此平台对各种服务品质进行测试，得到翔实的资料，为后继开发打下基础。

3.经济效益。ADSL 在国内 2002 年会发展到 900 多万个人用户[19]，按照每份 PPPoE 客户端软件 3 美元的估计，也可以节省将近 3000 多万美元。前面说过，PPPoE 软件的费用将被计入设备成本，客户端软件价格的降低会带动 ADSL 服务的降低，这有助于市场的开拓，吸引更多用户，从而再进一步降低 ADSL 接入的平均成本，有利于 ADSL 接入业务在我国的良性发展。

1.5.2 论文主要研究工作内容

本课题的目标是实现 PPPoE 客户端软件。所从事的工作内容主要有两大部分：一是对 PPPoE 协议的研究；二是 PPPoE 协议在 Windows 系统中的实现。重点在于实现 PPPoE 协议，而在 Windows 系统中实现该协议，则需解决以下几个关键性问题：

1. 确定 PPPoE 协议在 Windows 系统中的层次
2. 实现 PPPoE 协议同 Windows 之间的无缝结合
3. 把握 PPPoE 协议同 PPP 之间的关系
4. 在应用程序中实现对 PPPoE 协议细节的支持
5. 开发平台的搭建及开发环境的选择

所开发的 PPPoE 客户端软件将具有以下特性：

1) 功能和性能指标

开发的软件能够适用于国内大多数的家庭用户，能够适用于 Windows9x，WindowsNT，Windows2000 等操作系统。通过简洁友好的使用界面，普通用户能够正确连接到 ADSL 服务器。该软件应该能够通过 ADSL Modem 向用户提供稳定的 ADSL 宽带连接。

2) 软件运行环境

操作系统：Windows9x/NT/2000

硬件系统：P166；16M RAM；ADSL Modem；以太网卡；电话线

本课题的成果形式：

1) 软件概要设计和详细设计书
2) 软件测试报告
3) 系统配置和用户使用手册
4) 源代码

1.5.3 论文的结构

论文结构简述如下：

第一章概述，介绍 ADSL 和 PPPoE 在国内外的发展情况以及选题的意义；第二章介绍 PPPoE 协议的主要内容、Windows 系统下的网络组件的架构以及实现 PPPoE 协议所涉及到的技术，并通过分析给出实现 Windows 下 PPPoE 协议的方案；第三章介绍 PPPoE 客户端软件的结构；第四章详细阐述 PPPoE 协议驱动的实现；第五章总结论文的工作并对 PPPoE 技术进行展望。

本文的论述采用一般时序，在确定开发方案之后对软件各功能模块逐一描述。各功能模块是以软件的安装，初始化，到使用所涉及的顺序来进行介绍的。为了表述清晰，本文将开发过程中所涉及到的重点问题分散到各个子任务中，比如在 2.2.2 节确定 PPPoE 协议的层次及其同 Windows 组件间的关系；在 3.3.2 中讲述应用程序如何同核心驱动进行通讯；在 4.2 节中描述如何虚拟一个具有 WAN 和 TAPI 支持的网络适配器；在 4.5 节中讲述 PPPoE 对 DOS 攻击的防范等等。

第二章 PPPoE 协议及其实现的方案设计

2.1 PPPoE 协议

PPPoE 协议是将 PPP 帧封装在以太网数据报中的协议。PPP 是数据链路层协议，主要用来封装来自网络层的数据报，并将其通过串行线传输。

2.1.1 PPPoE 发现[15][26]

尽管 PPP 是点对点的协议，PPPoE 却是一个客户机/服务器的协议。客户端（通常是 ADSL 用户）查找 PPPoE 服务器（也称为接入多路复合系统, Access Concentrator，简称 AC）并且获得 AC 的 MAC 地址以及会话 ID。

一个 PPPoE 会话就是两个点对点端通过以太网通讯的过程，每一端都必须知道对方的网卡地址，而且，每个会话都必须用一个独特的会话 ID 来标志。

建立会话的这一过程即 *PPPoE 发现*，此过程中双方用以交换信息的以太网帧的类型都被设为 0x8863。

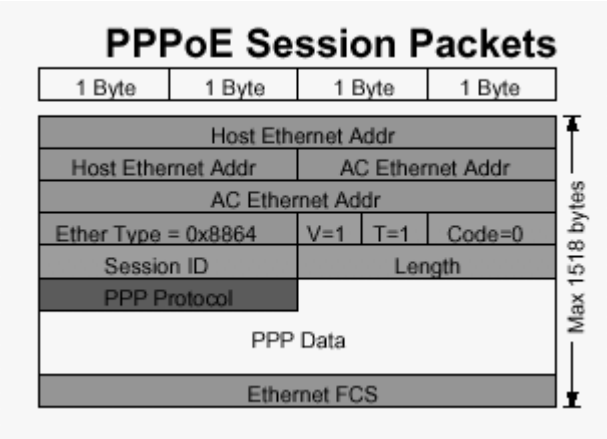
在*发现*过程中，PPPoE 客户端发送 PADI(PPPoE Active Discovery Initiation, 发现初始桢)，目标地址为广播地址，也可以在此帧中指定客户端所感兴趣的服务名称。AC（服务器）收到 PADI 后，以 PADO（PPPoE Active Discovery Offer,发现提供帧）回应，报告自己所提供的服务名称，这时目标地址就是客户端的 MAC 地址。

通常情况下，PADR 会来自一个以上的 AC。这时，客户端就需要选择一个它所要建立会话的服务器，并发送 PADR(PPPoE Active Discovery Request，发现请求帧)给该服务器的 MAC 地址。AC 检验该帧的合法性，继而分配 PPP 会话的资源，并指定同该客户通讯的会话 ID，然后将该 ID 在 PADS（PPPoE Active Discovery Session-confirmation，发现确认帧）中一并发给客户机。

客户机接受到 PADS 后，便知道了会话 ID 和服务器的 MAC，便分配 PPP 会话的资源，准备开始正常的 PPP 会话，进入 PPPoE 会话阶段。

2.1.2 PPPoE 会话

在 PPPoE 会话阶段，以太网帧的类型被设为 0x8864。图表 3 是此阶段的以太网帧的结构。



图表 3 PPPoE 数据帧（会话阶段）

此处的 PPP 数据不用填充，也不包括 PPP 的帧校验序列，这是因为以太网帧本身就有了校验序列。

在整个会话过程中双方通讯所用的以太网数据包都是发往对方网卡，而会话 ID 保持不变一直到会话结束。关于 PPPoE 的状态的转换请参见附录 1。

2.1.3 PPPoE 的优点

PPPoE 对接入商来说好处是显而易见的：

1. PPPoE 会话实质上是 PPP 会话，所以 IP 地址可以灵活分配，服务提供商可以在每次和用户建立连接时给用户分配 IP 地址。
2. PPPoE 在以太网上使用‘会话’的概念，服务商可以按连接时间计费。这样可以避免用户永久连接从而耗尽服务商有限的 IP 资源。
3. PPP 的会话需要用户认证，所要 DSL 服务提供商可以与选择的拨号网络 ISP 一样对用户正确收费而无需知道用户的确切位置。

同时对最终用户来说，通过 PPP 进行认证接入 Internet 这种方式并没有改变，用户界面保持不变，系统设置更加友好，可互操作性加强，更容易使用。

总之，PPPoE 这种解决方案，简化了用户的配置程序，又使用价格低廉的 Ethernet 网卡，而且它利用了目前广泛使用的标准拨号方式，对现有设备和操作系统进行尽量少的改动，实现了无缝整合，故而成为 ADSL 宽带提供者的首选方案。

2.2 Windows 系统下 PPPoE 的实现

2.2.1 OSI 参考模型与 PPPoE[20]

要实现网络协议 PPPoE，就需先确定其在 OSI 参考模型中所处的层次。

OSI 将整个网络的通信功能划分为七个层次，由低层至高层分别称为物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。其中物理层涉及在通信信道上传输二进制数位；链路层对传输的数据流产生和识别帧界；网络层关系到子网的运行控制；传送层从会话层接受数据；

会话层允许不同机器上的用户之间建立会话关系；表示层涉及所传送信息的语法和语意；应用层则主要是面向用户的应用程序。

物理层是 OSI 模型的最底层，完成物理介质上无结构的位流的接收和发送。它由电气/光，机械和物理介质接口三部分组成，负责为其上各层传送信号。在 Windows 系统下，物理层由网络接口卡（NIC，Network Interface Card）和与 NIC 相连的介质实现。对于使用串口的网络组件，它还包括将位流转换成包数据的低层软件。

OSI 将数据链路层分为两个部分：LLC（Logic Link Control）和 MAC（Media Access Control）。LLC 子层提供了点到点无错的数据帧的传输。按照服务的具体要求，LLC 分成三种类型：无连接的，带确认的和有连接的。现在用于网卡的 LLC 大多是无连接的，不需要处理重传和顺序发送。LLC 属于传输驱动程序。MAC 管理到网络介质的存取，检查帧错和识别地址。MAC 的功能由网卡的驱动来实现。

网络层控制子网的的操作。它根据网络条件，优先级和其它一些因素决定数据通过的路径。它的功能包括路由，网络流控，报文分片和重组，逻辑地址到物理地址的映射和计费。

传输层保证消息无错、有序的传送，没有数据丢失和重复。

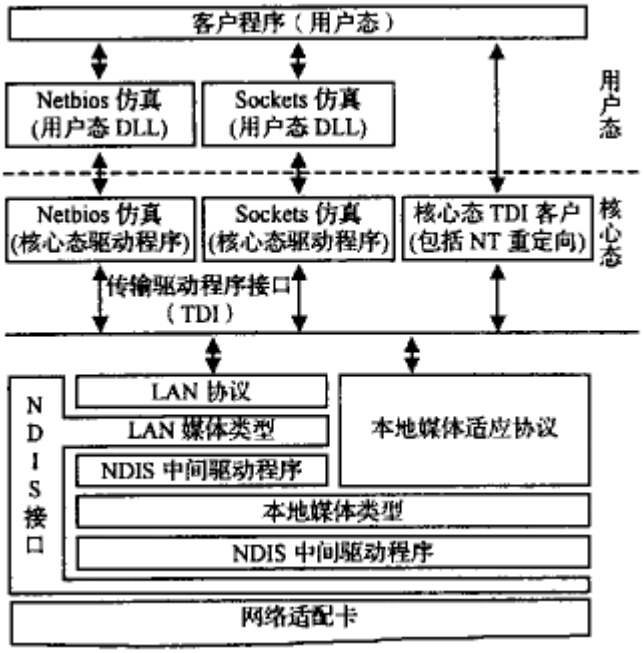
PPPoE 协议因为要封装 IP 包，所以它一定是处于网络层之下，又因为它要从以太网接口（网卡）发送数据而且具有设备无关性，所以它又应该在物理层之上。由于 PPPoE 要封装 PPP 数据帧，所以其所处的位置应该至少跟 PPP 协议的位置一样或者更低，众所周知，PPP 协议是数据链路层的协议。

通过以上的分析，可以得出结论：PPPoE 协议在 OSI 七层模型中应属于数据链路层的逻辑链路层。

2.2.2 NDIS 体系

NDIS（Network Driver Interface Standards,网络设备驱动接口标准）是开发 Windows 驱动所要遵循的规范。它描述了网络卡的驱动程序同上层协议以及操作系统通讯的接口，对驱动开发者提供一个抽象层，屏蔽了下层各种网卡的差别，同时又为上层的协议驱动提供服务。

NDIS 体系也是遵循 OSI 七层模型的，不过具体划分层次的时候，并没有一一对应的层次结构。NDIS 定义了物理硬件，小端口驱动，协议驱动，传输驱动接口四个层次。图表 4 是 NDIS 体系示意图。



图表 4 NDIS 体系示意图

NDIS 支持三种网络驱动：小端口驱动，中间层驱动，协议驱动。

小端口驱动：小端口驱动实现管理物理网卡所必需的硬件相关的操作，包括发送接收数据。但小端口驱动并不调用操作系统例程，而是通过调用 NDIS 提供的函数实现的。一个 NDIS 小端口驱动程序（也称为一个小端口 NIC 驱动程序）有两种基本功能：

A.管理一个网络接口卡（NIC），包括通过 NIC 发送和接收数据

B.与高级驱动程序接口，例如和中间驱动程序和传输协议驱动程序

一个小端口 NIC 驱动程序通过 NDIS 库和它的 NIC 及高层驱动程序相互通讯。NDIS 库导出一个完全的函数集合（NdisXXX 函数），来装入小端口需要调用的操作系统函数。然后，小端口必须导出一套 MiniportXxx 函数的实体指针，以供 NDIS 自己使用或代替高层驱动程序访问小端口。

下面用发送和接收操作举例说明小端口 NIC 驱动程序与 NDIS 和高层驱动程序之间的交互：

a.当一个传输驱动程序需传输一个数据包时，它调用一个由 NDIS 库导出的

NdisXxx 函数。然后 NDIS 通过调用适当的 NdisXxx 函数将这个数据包传至小端口。接着小端口通过调用适当的 NdisXxx 函数将数据包传至 NIC 来传输。

b.当一个 NIC 接到一个数据包时， 它可以发布一个硬件中断让由 NDIS 或 NIC 的小端口来进行处理。NDIS 通过调用适当的 MiniportXxx 函数来通知 NIC 的小端口。小端口对来自 NIC 的数据建立传输，然后通过调用适当的 NdisXxx 函数标识接收到的数据来绑定到高层驱动程序上。

协议驱动：上层协议驱动实现一个 TDI 接口或另一个应用程序定义的接口，来提供服务。这种驱动程序分配数据包，拷贝数据并通过调用 NDIS 将数据包发送至下层驱动。协议驱动同时也提供从下层驱动接收数据包的接口。

一个在 NDIS 驱动程序层级中处于最高级的网络协议经常在一个传输驱动程序中被用来做最低层的驱动程序来实现一个传输协议栈，如 TCP/IP 或 IPX/SPX 栈。一个传输协议驱动程序分配，拷贝来自发送应用程序申请发送的数据包，一个协议驱动程序同时也提供一个协议接口来接受从

下一个到达驱动程序中到达的包，一个传输协议驱动程序转移接受到的数据给适当的客户端应用程序。

在它的下端，是一个与中间网络驱动程序及小端口 NIC 驱动程序连接的协议驱动程序的接口。协议驱动程序调用 NdisXxx 函数来发送数据包，阅读并设置由低级驱动维护的信息，并使用系统服务。协议驱动程序也导出一套实体指针（ProtocolXxx 函数），可供 NDIS 自己使用或代替低层驱动程序标记收到的数据包，显示低层驱动程序的状态，并和其它协议驱动程序相互通讯。

在它的上端，传输协议驱动程序有一个用于同在协议栈中的高层驱动程序通讯的接口。

中间层驱动：中间层驱动位于协议和小端口驱动之间。在它的下端，一个中间驱动程序导出协议实体指针（ProtocolXxx 函数），让 NDIS 调用来完成与底层小端口的通讯请求。对于一个下层小端口驱动程序来说，一个中间驱动程序就像一个协议驱动程序一样。在它的上端，一个中间驱动程序导出小端口实体指针（MiniportXxx 函数），让 NDIS 调用来和一个或多个上面的协议驱动程序进行通讯。对于一个上层的协议驱动程序，一个中间驱动程序就像一个小端口驱动程序。

虽然它给它的上端导出一个 MiniportXxx 函数子集，一个中间驱动程序事实上并不能管理一个物理 NIC。实际上，它提供一个或多个虚拟适配卡，让上端的协议驱动程序可以绑定。对于一个协议驱动程序，一个通过中间驱动程序提供的虚拟适配卡就像一个物理 NIC 一样。当一个协议驱动程序向虚拟适配卡发送数据包或请求时，中间驱动程序传播这些数据包和请求给下层小端口。当下层小端口表示接收到数据包或响应这个协议的请求信息，或表明状态，中间驱动程序传播同样的数据包，回复和状态给绑定在这个虚拟适配卡上的协议驱动程序。

中间驱动程序通常以下几种方式使用：

1.在不同的网络媒介中起翻译的作用。如处于 Ethernet 和 Token Ring 传输层和一个 ATM 小端口中的中间驱动程序的功能是将 Ethernet 和 Token Ring 数据包映射给 ATM 数据包，反之亦然。

2.过滤数据包。一个数据包调度程序就是中间驱动程序用来过滤数据包的例子。一个数据包调度程序阅读由传输层传输的每个要发送数据包及由小端口标识的接收到的数据包中的优先信息。然后数据包调度程序就根据它的优先顺序输出或接收每一个数据包。

3.负载均衡。在多个 NIC 中平衡数据包传送。一个负载均衡驱动程序在传输协议之上表现为一个虚拟适配卡，但实际上是通过在多个 NIC 上分布的发送包，而具体在哪个 NIC 上发送，则将视各个网卡上的任务承载量而自动选择。

PPP 本是用来支持串口设备 modem，isdn，但是 PPPoE 则要求在以太网上建立 PPP，这涉及到网络媒质的转化，所以 PPPoE 协议驱动应该是属于中间层驱动程序这一类。

NDIS 体系中所涉及的层次颇多，所以概念比较多。中间层驱动的特殊性以及 NDIS 体系支持的重要性，在以下章节中会逐步展现出来。

2.2.3 Windows 2000 网络组件

Windows 2000 它具有下列特点：

1.Windows2000 是基于 NT 技术的，同以往 Windows 系统相比，包含了更多的网络支持，并且这些支持都是同 I/O 系统和 Win32 很好地整合到一起的。

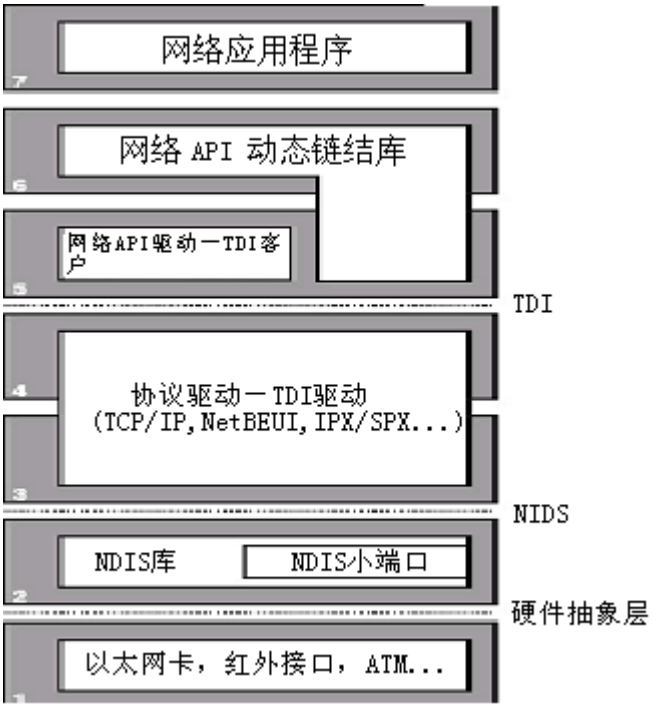
2.Windows2000 对开发者的支持很完善，而且驱动协议的开发由于环境的限制，向来就是采用先在 NT 系统上开发，再移植到 Win9X 系统的策略（Win9X 同 NT 的驱动是二进制兼容的）。

本文中如不加特别说明，默认的操作系统是 Windows2000 Professional.

2.2.3.1 Windows 网络组件[21]

Windows 的网络结构较好的适配了 OSI 七层概念模型，它在系统中完成最下面的四层的功能，这几层不为用户所见，但对网络起到支撑作用，而这几层也正是本文所需研究的对象。下四层通常被笼统地称为“ Transport ”，而上三层的组件都被称为“ Transport 用户 ”。

图表 5 是 Windows2000 的网络体系的略图，展示了每个网络组件在 OSI 参考模型中所处的层次，这种映射也并不是一一对应的，有很多组件都跨越了不同的层次。



图表 5 Windows2000 的网络体系

1)网络 API。向应用程序提供一种与协议无关的通过网络通讯的方式，可以在内核和应用态使用。
2)TDI 客户。内核态的设备驱动，是网络 API 的内核态的实现。
3)网络协议驱动（TDI Transports），是工作在系统内核的协议驱动。接收从 TDI 客户发送的 IRP，并且处理这些 IRP 代表的请求。这些请求也许需要增加一些与协议相关的头，例如(TCP，UDP，IPX)到 IRP 所处理的数据中，并且通过 NDIS 函数来与网卡通讯。
4)NDIS 库(ndis.sys)。Windows2000 对 NDIS 规范的支持组件，提供对网卡驱动的封装，隐藏 win2k 内核环境的特殊部分。
5)NDIS 小端口。是一些内核态的驱动，负责网络协议驱动与特定网卡的通讯，这种驱动并不处理 IRP，而是注册一个与 NDIS 库的接口，通过使用 NDIS 提供的函数来与网卡通讯，这些 NDIS 的函数会调用硬件抽象层(Hardware Abstraction Layer)的函数，完成实际的硬件操作。

Windows 系统的网络组件都对应 OSI 模型，比如物理网卡的驱动对应于物理层，TCPIP 的协议驱动对应于网络层，而 PPP 协议实现数据链路层的功能。每个组件的功能决定了它的网络层次，也就决定了它的实现方法。为了上下通讯，Win2k 中大量的网络组件按照逻辑层次被绑定在一起，这种绑定关系不仅表现在驱动的实现中，还表现在系统注册表中。例如 NDIS 的中间层驱动如果

要和 TCP/IP 或其他上层的驱动通讯，还要与下层特定的网卡通讯，就要建立它们之间的逻辑关联，才可以保证数据正确发送。而 NDIS 库作为一个编程的接口，接收驱动中的 NDIS 函数发送的消息，根据组件的逻辑层次决定发送的目的地。网络组件在注册表中也体现了逻辑层次，即绑定关系。例如物理网卡绑定 TCP/IP 协议，在物理网卡的注册表项目中就会有代表 TCP/IP 的信息。

2.2.3.2 WAN（广域网）环境

在 Windows 系统中，定义了三种网络驱动程序工作环境，无连接网络，面向连接网络，广域网络环境。

无连接网络，以太网，令牌环网络，就是无连接类型的；NDIS 中把无连接网络环境作为标准网络驱动程序环境，也就是说在 Windows 系统内部，传送的数据包将采用 NDIS_PACKET 格式（详见 DDK 文档）。

面向连接的驱动程序，比如 ATM。

广域网络环境，它定义了点到点连接，支持无连接和面向连接的媒介之上的 WAN（广域网）连接，所使用的硬件设备可以是任何无连接网络设备或者有向网络设备或其他专有设备。由于 PPPoE 协议的实现涉及到拨号，PPP 等 Wan 的概念，所以在此重点讨论一下这种环境。

WAN 环境包括下列部分：RAS，TAPI 服务供应者，NDISTAPI，TAPI 代理器，NDISWAN，WAN 小端口和串行驱动程序。

其中 RAS 允许用户模式应用来建立拨号连接；TAPI 服务供应者是一个用户模式组件，它根据来自 RAS 客户和其他的应用程序，通过服务提供者接口（SPI）来接收调用设置和分片的请求。TAPI 服务提供者负责将对无连接媒体的调用申请发送给 NDISTAPI，而将对面向连接的媒体之上的调用申请发送给 TAPI 代理器。NDISTAPI 是实现无连接小端口到 TAPI 驱动的核心模式组件；TAPI 代理是用来实现面向连接小端口到 TAPI 驱动的核心模式组件；串行驱动程序是一个标准的设备驱动程序。

WAN 小端口：一个 WAN 小端口和一个非其他小端口驱动一样，可以调用多个同样的 NDIS 函数，并提供多个相同的处理程序。但是当发送包和显示收到数据包给上层协议时，WAN 小端口会调用与 WAN 相关的 NDIS 函数。WAN 小端口也使用一个 NDIS_WAN_PACKET 的结构替换 NDIS_PACKET 结构。另外，WAN 小端口维护 WAN 相关的信息并响应 WAN 的这一信息的相关请求。一个 WAN 小端口可以支持无连接或面向连接的媒体，一般而言，WAN 小端口驱动都由硬件设备商提供，比如 isdn 卡，56k 调制解调器等拨号设备。

NDISWAN 是一个实现 PPP 协议，链路成帧，压缩和加密的 NDIS 中间层驱动。NDISWAN 将上层传输驱动程序的数据包从局域网格式（NDIS_PACKET）转换成城域网格式（NDIS_WAN_PACKET），并将重新格式化的数据包传给下层 WAN 小端口驱动程序，在转换数据格式时，NDISWAN 用以简单的 HDLC 帧格式。

在 Windows 系统中，NDISWAN 是以 ndiswan.sys 存在的，它作为一个协议绑定在拨号适配器上，同时又作为一个小端口被上层传输协议（如 TCP/IP, IPX/SPX）所绑定，以 TCP/IP 数据包的传输为例，数据的传输方向和数据格式为：

TCP/IP←(IP 包)→NDISWAN←(PPP 封装包)→小端口。假如系统以普通 56k 的调制解调器拨号上网的话，此处的小端口就是 modem 的驱动程序了。

参照 2.1 节对 PPPoE 协议的分析，PPPoE 协议要做的是将 PPP 的数据包转换为 PPPoE 封装后的以太网数据包，这一过程同 NDISWAN 所做的很相似，甚至可以说只是转换的方向相反。因为 NDIS 向 NDISWAN 传下来数据时，仍视 NDISWAN 为下层的小端口，所以数据格式也是 802.3，而只不过数据包的目标地址和源地址并非实际的 MAC 地址而已。（这个结论可以通过截获系统

发往拨号网络适配器的数据得到，这一工作可以利用 Windows2000 Sever 下的网络监视器来进行。）

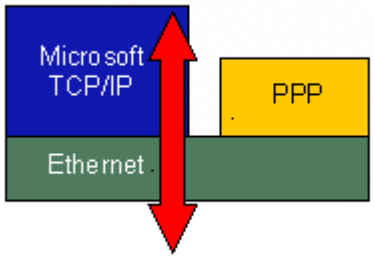
所以，假如在 NDISWAN 之下再加一层中间层驱动，来实现 PPP 到 PPPoE 的转换，应该可以做到在以太网上的点到点传输。因为 NDISWAN 本身就是处与逻辑链路层的驱动，所以采取这种方案，所要实现的驱动必然在逻辑链路层，这也验证了本章 2.2.1 节的结论。

2.3 实现方案的确定

2.3.1 两种方案的协议结构

制定在 Windows 系统中具体实现 PPPoE 的方案时，先后提出并尝试过以下两种方案：

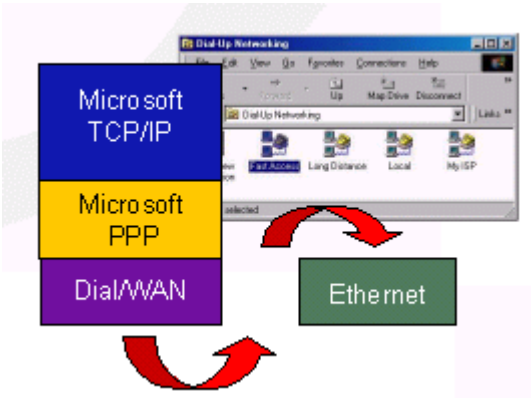
1. 在以太网上直接实现 PPP，协议结构如图表 6。



图表 6 直接实现 PPPoE

这种方案正是 Enternet 所使用的方式。在虚拟网卡上直接实现其他协议到 PPP 协议的转换，绕过了 NdisWan，这是最为节省系统资源的。但是这种方案又需要具体地实现 PPP 协议，尽管这样可以提供更加灵活的控制，但是为了实现 PPPoE 而要去实现比它本身更加复杂的 PPP 协议，其实就是再去实现 NDISWAN 的功能，未免有些得不偿失。

2. 利用 Windows 系统中的拨号网络，只进行 PPP 到 PPPoE 的转换，协议结构如图表 7。



图表 7 利用 Windows 拨号网络组件实现 PPPoE

这种方案的好处在于，无需实现 PPP 协议的细节，直接利用 Windows 的拨号网络组件。因为这样实现 PPPoE，充分地利用了现有的 Windows 资源，而为软件的具体实现简化了很多。另外

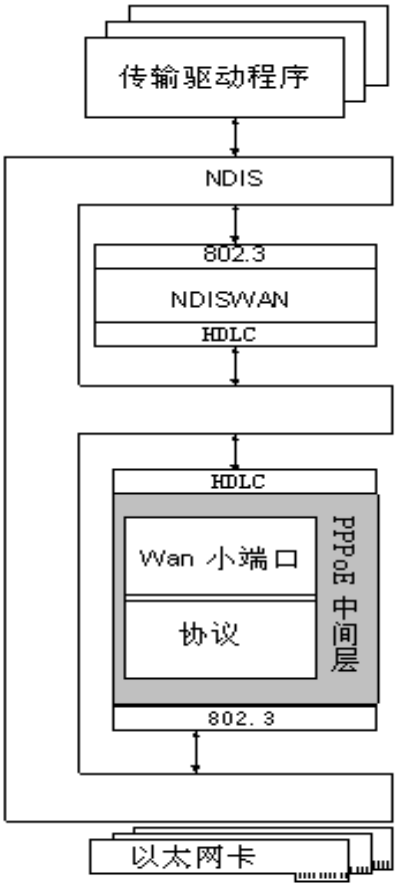
从效率上讲，已经完全可以满足 10M 的 ADSL 传输要求。

2.3.2 本文所采用的方案

通过以上的分析，本文将方案确立为：

利用 NDIS 中间层驱动程序机制，实现一个中间层驱动程序，为了同 NDISWAN 通讯（也就是说必须让 NDISWAN 绑定它），它需具有 Wan 小端口的上沿接口；为了同以太网卡通讯（也就是说必须可以绑定在以太网卡上），它需具有协议下沿接口，使其工作在 NdisWan 和以太网卡驱动程序之间，完成 PPPoE 协议转换工作；利用 WAN 小端口提供的上沿函数支持拨号网络的拨号服务，完成虚拟拨号。

在 NDIS 系统中，本协议驱动所处的位置如下图表 8所示。



图表 8 协议驱动在 NDIS 体系中的层次

本文在 2.2.3 节中介绍过 Windows 2000 的特点，考虑到本系统将被移植到大多数 Windows 系统之上，本方案将采用以下开发环境：

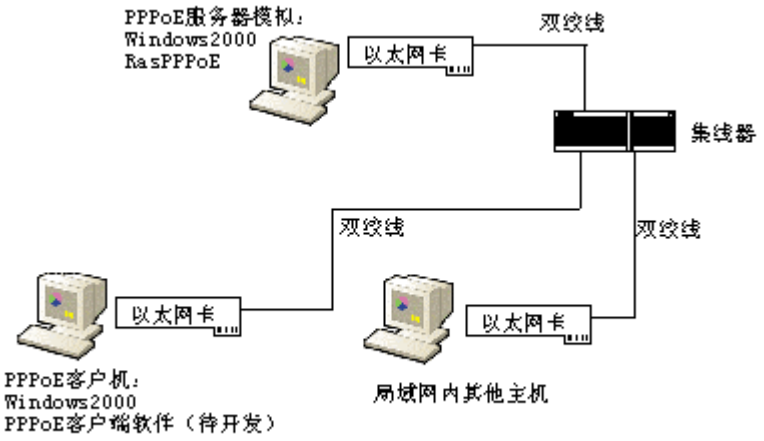
- 操作系统：Windows 2000 Professional
- 开发工具：Microsoft Visual C++ 6.0，Windows 2000 DDK
- 调试工具：Softice

C/S 模型的搭建：本课题曾试图在 Unix 系统上构造服务器模型，但是一直没能找到合适的服

务器软件。所以开题之初不得已决定先开发一个 PPPoE 服务器软件，然后再开发 PPPoE 客户端软件，可想而知，在两端都未确定的情况下，工作量将是相当大的。

Robert 于 2000 年 12 月发布的 RasPPPoE v0.95 新增加了服务器功能（参见 1.4.2），使得在 Windows 局域网上搭建服务端模型成为可能。由于 PPPoE 协议本身的硬件无关性，完全可以在两台 PC 之间直接实现 C/S 结构，而无需 ADSL 设备，这一模拟环境可以满足客户端软件的开发调试等工作需要。

图表 9 是在局域网内模拟 PPPoE C/S 的示意图。其中，作为服务器（AC）的主机运行 Windows2000 系统，安装了 RasPPPoE，并且在拨号网络中设置了接受拨入；客户机上是开发环境，用于软件的开发和调试。



图表 9 具有 PPPoE 模拟服务器的局域网开发环境

第三章 PPPoE 客户端软件

3.1 软件的总体结构

客户端软件包括三个部分：通知对象（Notify Object）、协议驱动程序（Driver）以及用户工具（Utility）。通知对象主要用于协助驱动程序的安装、卸载以及属性页的管理，存在形式为 DLL 文件；驱动程序实现 PPPoE 协议，完成数据包的转换，以 SYS 文件的形式工作；Utility 主要用于管理拨号网络地址簿，是一个 Win32 应用程序。另外一个重要组成部分是操作系统注册表，它记载了协议驱动的位置，操作层次以及协议运行所需的参数。

由于本方案是利用 Windows 的拨号组件，所以用户界面仍然是传统的拨号网络界面，这是此方案的另一个优点，也即最终用户不需再花费时间熟悉新的操作流程，同操作系统的无缝结合使得本系统的受欢迎程度极大地加强。本协议的安装和卸载跟其他协议并无不同，都需要调出网络属性，各种数据都存贮在 Windows 系统注册表中。

图表 10 是本系统各模块的关系一览。该图以字母顺序说明了本系统从安装到正常运行所发生的一般事件，椭圆形代表事件起点，箭头与字母表示了某事件发生时的数据流向，而箭头两端则是参与该事件的系统组件。

（a） 安装/卸载：协议的安装与卸载。

（b） 修改属性：改变协议参数的设定，比如 MTU 的默认值，最大的连接线路个数等。

（c） 协议初始化，虚拟网卡：绑定协议在物理网卡上，同时向系统虚拟出可拨号的网卡。

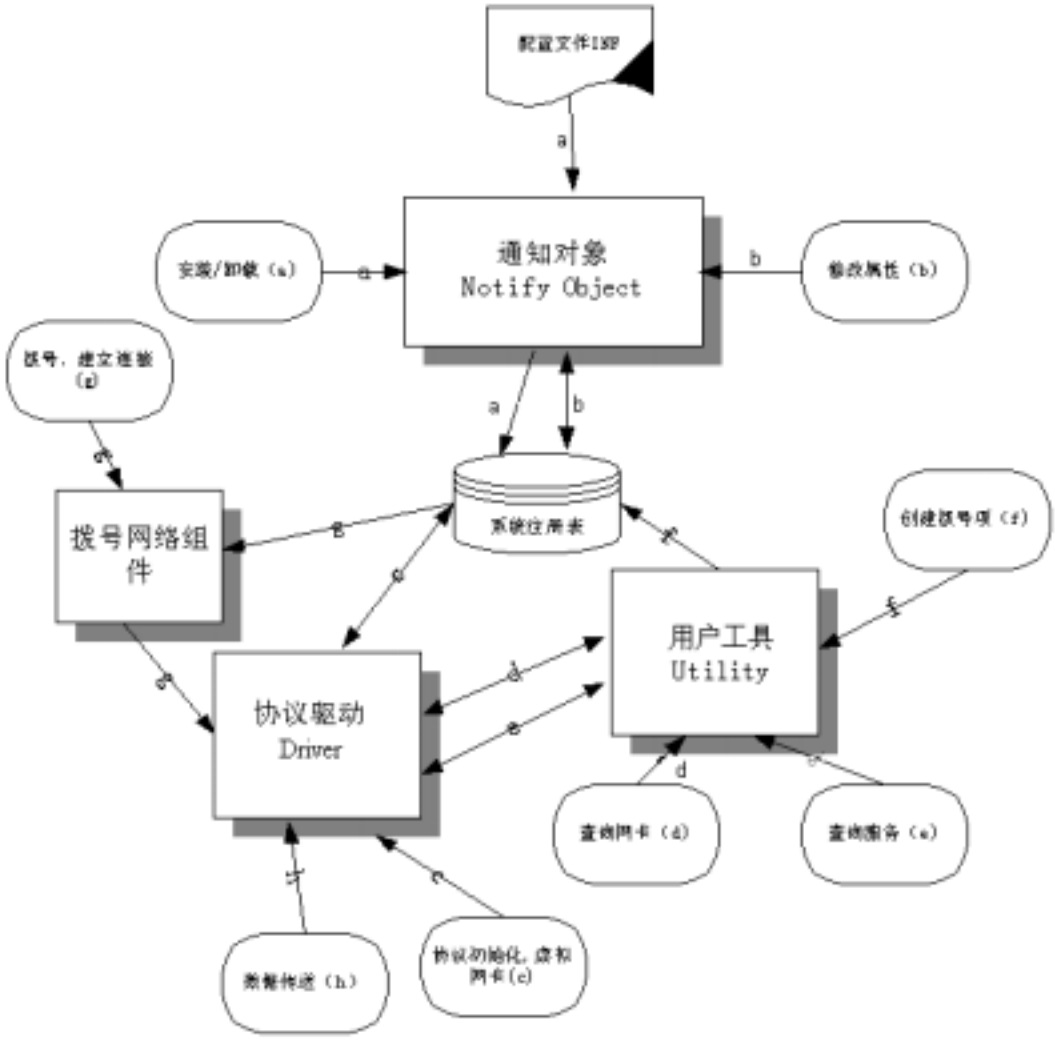
（d） 查询网卡：检测系统中所安装的网卡。

（e） 查询服务：查询网络中存在的 ADSL 接入服务。

（f） 创建拨号项：为网卡和通过该网卡所查询到的服务建立拨号网络的连接。

（g） 拨号，建立连接：同远端的接入服务器（AC）建立连接。

在系统的三个组成中，最为重要的是驱动程序，本章将主要介绍通知对象和用户工具等组件，驱动程序及其中涉及到的技术难点将在下一章中详述。

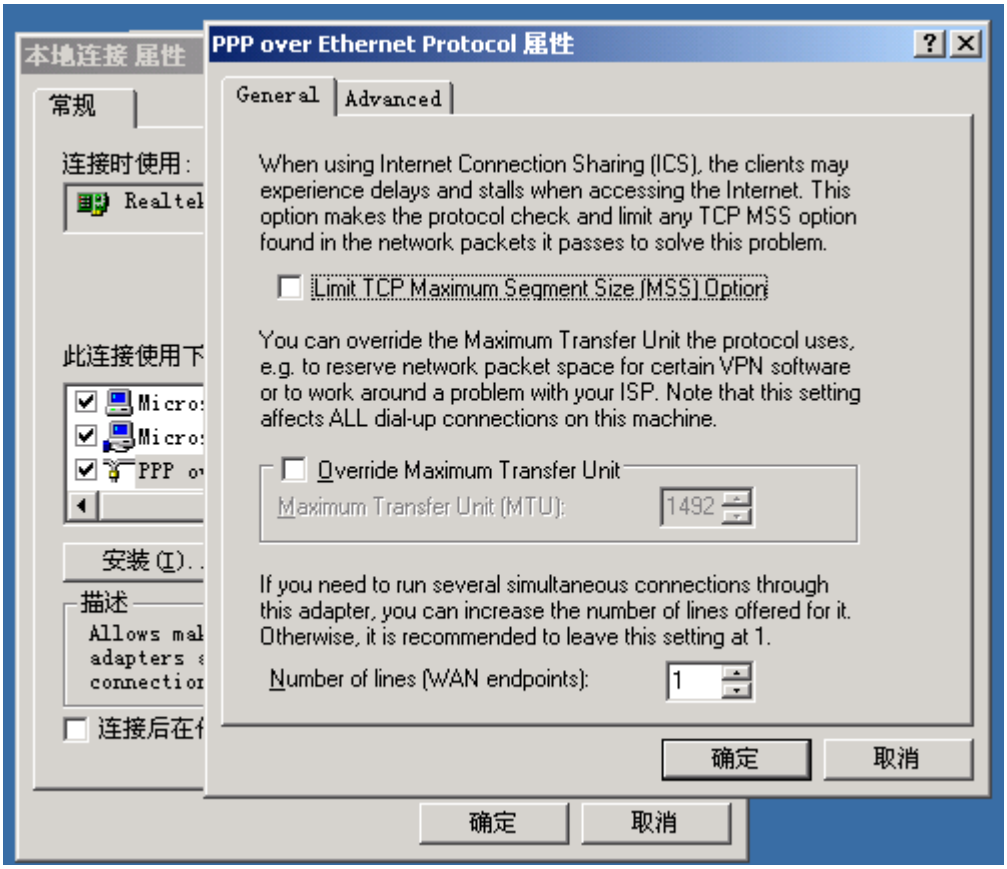


图表 10 系统结构图

3.2 通知对象（Notify Object）

通知对象是一个动态链接库，本系统软件包中的文件 pppoe.dll 就是通知对象，它主要负责处理从网络配置子系统发送至 PPPoE 协议驱动的通知，完成协议的安装和卸载，并显示用户界面，让用户能够对协议的属性配置进行控制。

通知对象的功能比较简单，主要是用操作系统定义好的 COM 接口实现的，本文不做详细介绍，这里只给出利用通知对象实现的用户界面（见图表 11）。



图表 11 修改协议参数的用户界面

3.3 用户工具（Utility）

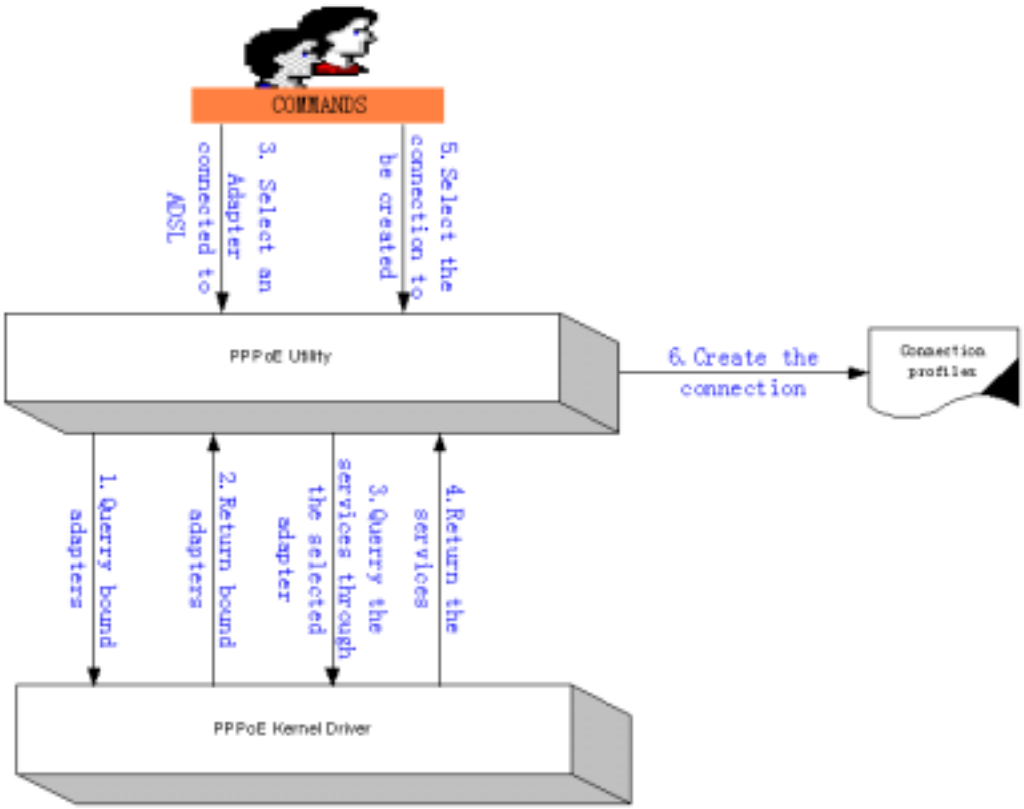
PPPoE Utility 是 PPPoE 客户端软件的一个组件，它利用 PPPoE 协议的功能，查询目前可用的服务名称，并为用户指定的服务建立连接项；同时它又可以作为一个诊断工具，如果运行正常，则说明 PPPoE 协议在本机上正常安装，反之，则需要重新安装。

PPPoE Utility 最终文件名为 PPPoE.exe，并运行在 **Windows2000** 系统之上。

3.3.1 程序系统的结构

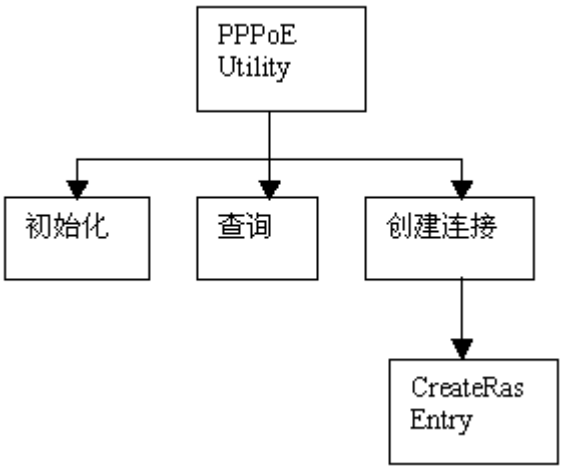
PPPoE Utility 属于应用程序，运行在用户区，而 PPPoE 协议处于系统核心态，所以二者之间的通讯需要提供系统文件来进行。

PPPoE Utility 同协议驱动以及用户之间的控制流程见图表 12。



图表 12 控制流图

PPPoE Utility 是基于对话框的 Win32 程序，主要功能都集中在 CDialog 的派生类的成员函数中。图表 13 是 PPPoE Utility 的结构以及模块间相互调用关系：



图表 13 用户工具结构

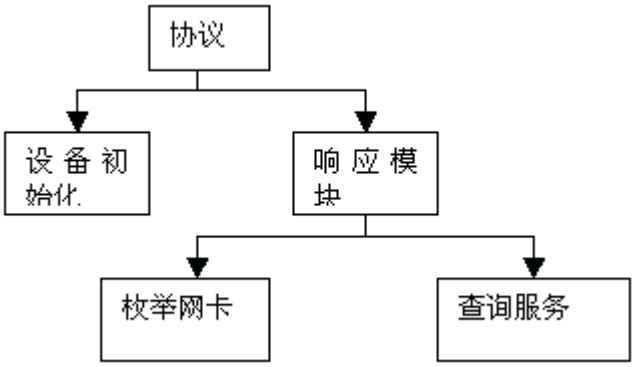
各模块所对应的子函数的名称：

初始化－OnInitDialog

查询－OnButtonQueryService

创建连接－OnButtonCreateConnection

相应地，协议驱动也有功能模块来响应从应用程序传送过来的命令。图表 14 是驱动程序中的响应模块。



图表 14 协议的响应模块

各模块所对应的子函数的名称：
设备初始化：DriverEntry()
响应模块： PPPOEIoControl（）
枚举网卡：PPPOEGetAdapterList（）
查询服务：PPPOEQueryServices（）

3.3.2 应用程序同驱动程序之间的数据交换

本协议驱动程序将在系统中虚拟出来拨号网卡，所以应用程序同驱动之间的数据交换被系统视作为对设备的操作。

应用程序对设备的存取通过提交 IRP(Input/Output Request Packet 输入/输出请求包)来进行。应用层的调用象 CreateFile,ReadFile 等等将导致系统的 I/O 管理器生成一个与应用层的调用相对应的 IRP。在操作系统中，对设备的存取过程是这样的：操作系统中的 I／O 管理器接受 I／O 请求（通常是由用户态的应用程序发出的），建立相应的 IRP 来描述它，将 IRP 发送给合适的驱动程序，然后跟踪执行过程，当操作完成后，将返回的状态通知请求的发起者。操作系统中的 I／O 管理器、即插即用管理器、电源管理器都使用 IRP 来与内核模式驱动程序、WDM 驱动程序进行通信，并且，各驱动程序之间的通信也是依靠 IRP[22]。

在各层驱动中，IRP 首先从最上层进入，然后，依次往下传送。在每一层，驱动程序自行决定对 IRP 的处理。一个驱动程序可以只继续 IRP 向下传递，而并不做任何事情，也可以完全接管 IRP，不再把它向下传递了。当然，驱动程序也可以处理 IRP 后，再把它继续向下传递。这取决于驱动程序的功能和 IRP 的含义。在我们所实现的驱动中，只要判断出来是发送给自己的 IRP，就会对其进行处理，然后便返回，没必要再传送到下层去了。驱动将根据应用程序发送来的 IRP 得到命令控制号（IoControlCode），据此控制号进行分支处理。在 PPPOEIOControl（）中所承认的有效命令控制号只有 IOCTL_ENUM_ADAPTERS 和 IOCTL_ENUM_SERVICES， 响应模块分别转入函数 PPPOEGetAdapterList 和 PPPOEQueryServices。
命令控制号的定义语句为：
#define IOCTL_ENUM_ADAPTERS CTL_CODE(FILE_DEVICE_PROTOCOL, 3 ,

METHOD_BUFFERED, FILE_ANY_ACCESS)
#define IOCTL_ENUM_SERVICES CTL_CODE(FILE_DEVICE_PROTOCOL, 4 , METHOD_BUFFERED, FILE_ANY_ACCESS)
CTL_CODE 是一个宏命令，在文件 devioctl.h 中定义，在应用程序和驱动中对命令控制号的定义必须一致。

在驱动的入口 DriverEntry()处，驱动向系统注册设备，同时注册为应用程序提供的函数的入口 PPPOEIOControl()和 PPPOECREATE()。

在注册 Miniport 之后，DriverEntry 调用

NdisMRegisterDevice(WrapperHandle,
&DeviceName,
&SymbolicName,
MajorFunctions,
&deviceObject,
&ndisDeviceHandle) 来注册设备，该函数用来创建一个设备对象（DeviceName），并创建一个该对象符号链接（SymbolicName），使用户得以通过该符号链接对设备进行操作。设备对象**必须**具备“ \\Device*DeviceName* ”的格式，而对象符号则需有“ \\DosDevices*SymbolicName* ” 的格式；在此处，*DeviceName* 和 *SysmbolicName* 均设为“ PPPOE ”。

MajorFunctions 的定义在

PDRIVER_DISPATCH MajorFunctions[IRP_MJ_MAXIMUM_FUNCTION+1];

该数组指定了对设备进行操作的各函数的入口地址，目前只定义了 PPPOECREATE、PPPOECLOSE 和 PPPOEIoControl。其中前二者是应用程序调用 CreateFile 和 CloseFile 时启动的，后者则是处理应用程序发来的其他控制命令，并转入相应的子函数。

3.3.3 全局数据结构和全局概念

HANDLE hFile
描述：为了实现用户程序同核心驱动程序之间的通讯，必须得到协议安装时所创建的设备的文件句柄。此变量即保存该句柄。
CComboBox m_comCtlAdapter
描述：复合框对象变量，用于保存并显示系统中所有的以太网卡的名称，再确切一点说就是 PPPoE 协议所绑定的网卡的名称。使用户可以选择 ADSL modem 所连接的那个以太网卡，然后针对特定的网卡进行查询。
ListBox m_listServices
描述：列表框对象变量，用于保存并显示对 ADSL PPPoE Server 的列表，然后用户可以选择对哪些服务建立连接。
ULONG NumAdapters
描述：保存系统中 PPPoE 协议所绑定的网卡的数目。
应用程序同驱动程序通讯时数据缓冲区之中数据格式的约定。

为了便于程序实现，在应用程序和驱动程序之间必须达成以致的约定，此处约定传送的数据均为 ANSI 类型，且各字符串之间以 NULL 相间隔。
关于服务商和服务名称的约定。

同驱动所默认的拨号对象的格式一致，由协议所返回的服务商和服务名称采取以下结构：
"Server_01" \\ "Service_01"(以空结尾)

"Server_02" \\ "Service_02"(以空结尾)
"Server_03" \\ "Service_03"(以空结尾)
PPPoE Utility 应当根据此约定处理所返回的数据。

协议名称以及所创建设备的名称的约定。

在驱动的入口函数 DriverEntry 中，协议被注册为“ PPPOE ”，而设备名称注册为“ \\DosDevices\\PPPOE ”，这样，在 Utility 中就必须以"\\\\.\\PPPOE" 为文件名来创建这个设备的文件句柄。

3.3.4 模块功能及其简要描述

3.3.3.1 初始化模块

功能描述：

创建设备文件，得到网卡列表

性能要求

由于此成员函数在对话框出现之前被调用，故应包含一切异常处理：文件无法创建，驱动所绑定的网卡数目为 0，并将相应信息在退出程序之前提示给用户。

接口

1)输入/输出：无参数。
2)所用到的全程数据项：NumAdapters，hFile
3)调用的函数：CreateFile（）建立同驱动程序的联接，并将句柄保存至 hFile 中，再调用 DeviceIoControl，向驱动发送 IOCTL_ENUM_ADAPTERS 命令。返回的数据缓冲区将保持以下结构：

ULONG NumberOfAdapters
LPTSTR Adaptername_1;(以空结尾)
LPTSTR Adaptername_2;(以空结尾)
..
..
LPTSTR Adaptername_NumberOfAdapters;(NULL terminated)
NULL;(此处的空值代表有效数据的结束)

驱动的响应支持

在安装协议时，驱动会为它所绑定的每个网卡分配一个描述块（包括网卡名称，绑定的上界和下界和各种其他参数），并将所有这些描述块组成一个链表。在响应应用程序对网卡的查询时，驱动会遍历这个链表，计算网络适配器名称长度的总和。若该总和大于应用程序提供的缓冲区的长度，则返回 STATUS_BUFFER_TOO_SMALL;反之，则按照上文所约定的格式将网络适配器的个数和名称保存在缓冲区内，返回 STATUS_SUCCESS。

应用程序得到已被正确填充的缓冲区后，便解析出网卡的名称并显示给用户，令用户选择。

3.3.3.2 查询模块

功能描述

根据用户选择，向指定网卡发送查询命令，分析返回的数据，得到服务器/服务名列表，并将结果显示。

性能要求

因为驱动要同网络之间有数据的交换，所用操作会有一个时间，故速度上受限于驱动程序的超时设置。但是应该是用户可以接受的。

接口

1)输入/输出：无参数

2)所调用函数：DeviceIoControl 向驱动程序发送 IOCTL_ENUM_SERVICES 命令，DeviceIoControl 的参数中的输入缓冲区与输出缓冲区是同一个，而输入缓冲区的第一个 ULONG 的空间贮存用户所选择的网卡的序号序号。(0...NumberOfAdapters)。因为传送的是序号，故在 PPPoE Utility 的界面上显示的网卡的顺序要同驱动所返回的顺序一致。

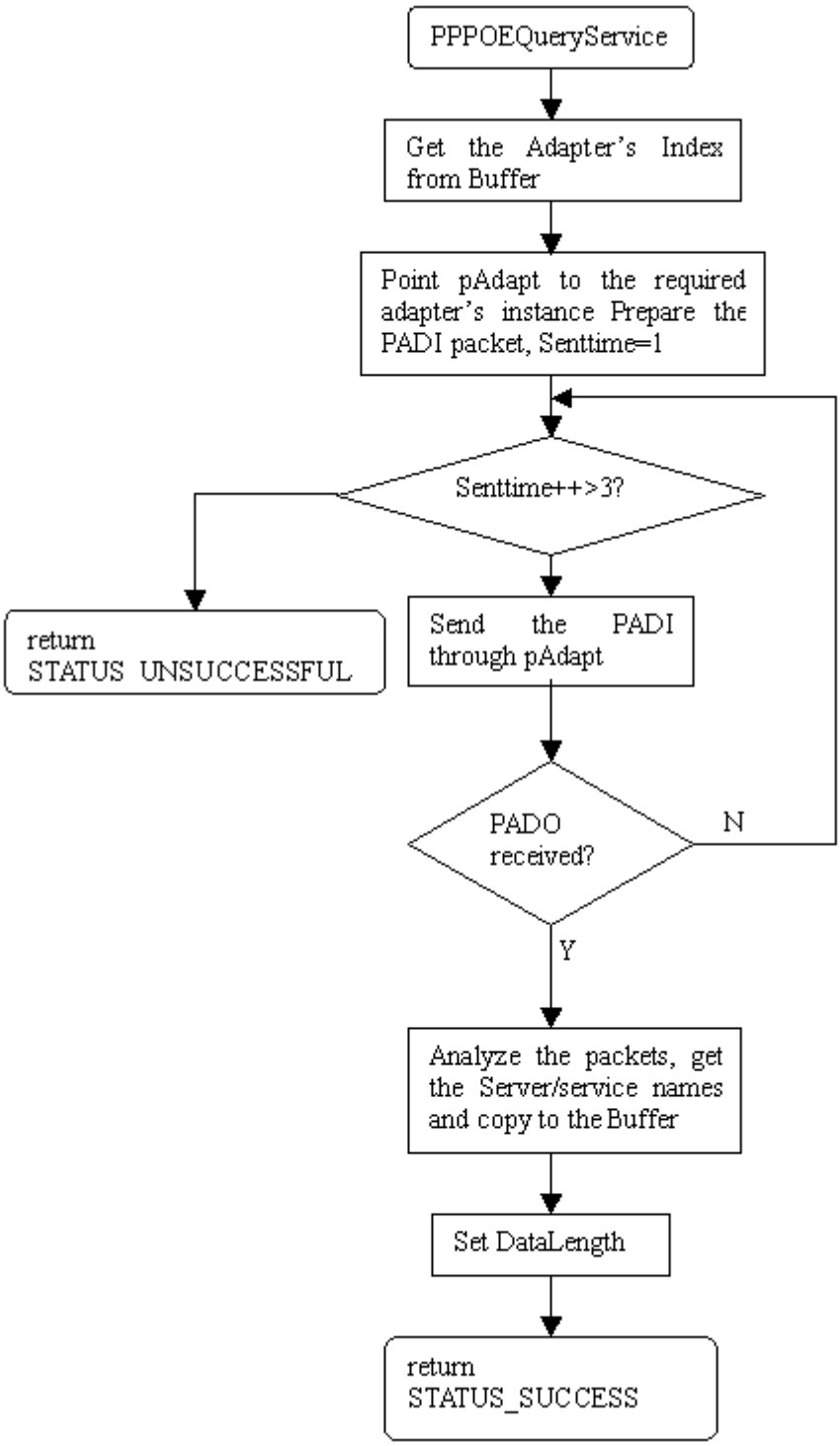
驱动的响应支持

在协议驱动层，接收到用户所指定的网卡的序号之后，在该网卡上发送 PADI 数据包，等待 ISP 服务器的 PADO，得到各服务器/服务的名称，并返回。

因为此过程要等待网络中服务器的响应，所以需要引入一个超时机制，以免造成应用程序的长时间没有得到查询结果而出现死锁现象。在驱动程序中是利用 NdisResetEvent/NdisWaitEvent 的联合使用而够成这个超时机制的，这是对单次查询起一个限时的作用。

另外，由于网络状况的不同，有时候会有丢失数据包的现象，也就是说虽然服务器按照 PPPoE 协议对 PADI 回应了 PADO，但是用户机器却没有收到。针对这种情况，本文采取了在规定时间内如果没有得到响应，那么就重新发送 PADI 数据包，而重新发送的次数限定为 3。

图表 15 是协议驱动对于用户工具查询服务的响应模块的流程，其中涉及到 PPPoE 的一些操作，如 PADI 数据包的生成，PADO 数据包的监测等将在下一章中再作介绍。



图表 15 协议对查询服务的响应

3.3.3.3 创建连接

功能描述

根据用户的选择，建立相应的拨号连接项。用户可以仅指定网卡，也可以同时指定服务器/

服务名称。如果是前者，则拨号时，系统自动连接到第一个返回响应的服务器的默认服务；而若是后者，系统将同所指定的服务器的指定服务进行连接。

性能要求

全部操作都在应用程序层，性能不作特别要求。

接口

调用全局函数 CreateRasEntry，传递参数网卡名，连接名和服务器/服务名称。

CreateRasEntry 判断接受的参数，调用 Windows TAPI 函数建立连接。

第四章 PPPoE 驱动程序的实现

4.1 PPPoE 驱动程序的功能划分

按照设计方案，PPPoE 驱动程序是一个带有 Wan 小端口的中间层驱动，所以从层次上划分，驱动分为两层：下层协议驱动和上层微端口驱动。

协议驱动主要实现同下层网卡的数据通讯，接收从网卡传来的以太网数据包，并将 PPPoE 包转换为 PPP 的数据，然后上传到 NDISWAN。

而微端口驱动则负责接收从 NDISWAN 下传来的 PPP 数据包，并将其转换成 PPPoE 格式，然后通过网卡发送出去。此处的下层协议驱动并非本文所讨论的 PPPoE 协议实现的全部，相反，PPPoE 协议的实现是在整个中间层驱动中的这两层配合下完成的。为了便于理解，本文从功能上对整个协议的实现做一介绍，同时将一些技术问题融合到其出现的模块中详做探讨，至于一些程序层面的问题则不做太多描述。

从功能上划分，则 PPPoE 驱动又分为：WAN 支持部分、TAPI 支持部分、链路控制、PPPoE 支持、发送/接收机制、同应用程序接口机制。

4.2 WAN 支持

为了接收 NDISWAN 下传来的 PPP 数据帧，首先应该令系统将我们的微端口视作拨号设备，也就是说需要虚拟出来一个拨号适配器来让 NDISWAN 绑定，这一步主要是在微端口初始化时完成的。

在驱动入口处 DriverEntry()，驱动在调用 NdisIMRegisterLayeredMiniport 时，要指定正确的 NDIS 版本，有关该 WAN 微端口各种函数的入口，包括端口初始化，查询/设置，微端口重置，微端口停止，数据发送等。

4.2.1 NDIS 版本的确定

NDIS3.0，4.0，5.0 都支持标准 WAN 小端口，而 NDIS5.0 则同时支持面向连接的 WAN 小端口驱动。

对一个实用的，面向多平台的协议驱动来说，NDIS 版本并非越高越好。事实上，Windows98 支持 NDIS4.0 以下，而 Windows2000 支持 NDIS3.0 到 NDIS5.0。但是在 Windows2000 中，尽管 NDIS5.0 提供支持，系统并不支持由 NIDS5.0 封装的标准 WAN 小端口，它只会盲目地认为 NDIS5.0 提供的 WAN 小端口是面向连接的，所以会在注册时调用相应面向连接的函数。这时，因为本文所要实现的协议并非是面向连接的，所以没有对这些函数的支持，便会引起系统崩溃。

要解决这个问题，只有用 NDIS4.0 或在驱动中提供对面向连接的支持。在 Windows2000 中，一个中间层驱动并不能提供面向连接的上端（Upper Edge）接口来同 NDISWAN 通讯，故只有使用 NDIS4.0。

4.2.2 微端口初始化

在驱动程序注册了 WAN 微端口后，NDIS 会调用微端口的初始化模块来让其完成必要的初始化操作，所传递的参数包含了系统所支持的介质类型数组（MediaTypeArray），全局信息句柄，绑定上下文句柄。

微端口可以通过全局信息句柄得到全局变量的指针，然后将其中有关 WAN 的成员填充，以待在系统对微端口的属性进行查询时使用。

另外需返回本微端口所属的介质类型在类型数组中的位置。MediaTypeArray 里面包含所有系统所支持的介质类型，这些类型在 DDK 中定义，Windows 支持的介质有 13 种之多，此处需要我们指出的有：

NdisMedium802_3

以太网卡(802.3).

NdisMediumWan

WAN 接口卡，这一类型包括了很多点对点和 WAN 接口卡，以及一些必须通过协议驱动和底层驱动协商才能确定数据格式网络接口。

如：Hub，X_25，ISDN，Serial，FrameRelay，ATM，Sonet，SW56K，PPTP，L2TP，Irda，Parallel。

一般的网卡都会返回 NdisMedium802.3，这样系统就将微端口当做以太网卡。而我们这里需要虚拟出来可拨号的网卡，所以在此必须选择介质类型为 NdisMediumWan，而且在后继的系统查询中，要选择 WAN 子类型 NdisWanMediumIsdn，如果安装成功，系统将会把我们的微端口当做一个 ISDN 卡，它就会在系统设备表呈现出 isdn 的属性，从而能够被 NDISWAN 所绑定。

4.2.3 查询/设置

调用微端口所注册的标准接口函数是系统在特定的事件发生时进行的控制方式，如初始化微端口，有数据需要发送等。系统同微端口通讯的另一种方式就是查询和设置。

上层驱动调用 NdisRequest 来查询或设置下层微端口的各种属性，这一请求经过 NDIS 系统的处理，到达微端口时，就分别成了 MiniportQuerryInformation 和 MiniportSetInformation，微端口通过 NDIS 传过来的 OID（Object Identifier，对象标志）来确定该查询或设置的类型以及其他参数的类型。

如当待查询的 OID 是 OID_WAN_MEDIUM_SUBTYPE 时，微端口就明白它需返回一个 WAN 的子介质类型，此处为 NdisWanMediumIsdn，并将此值填入被当做调用参数的输出缓冲区中。

当 WAN 微端口初始化之后，系统会发来 OID_WAN_GET_INFO 的 OID 来查询相关的 WAN 的信息。则 WAN 微端口就需填充一个 NDIS_WAN_INFO 的结构，来报告自己的属性，在 DDK 中，NDIS_WAN_INFO 的定义如下：

```
typedef struct _NDIS_WAN_INFO{
    OUT ULONG                   MaxFrameSize;
    OUT ULONG                   MaxTransmit;
    OUT ULONG                   HeaderPadding;
    OUT ULONG                   TailPadding;
    OUT ULONG                   Endpoints;
    OUT UINT                     MemoryFlags;
    OUT NDIS_PHYSICAL_ADDRESS  HighestAcceptableAddress;
```

```
    OUT ULONG                   FramingBits;
    OUT ULONG                   DesiredACCM;
} NDIS_WAN_INFO, *PNDIS_WAN_INFO
```

MaxFrameSize 是该驱动最多能发送和接收的数据帧的字节数，通常是 1500 字节，然而驱动应该能够处理比它汇报的最大数目还要多 32 字节的。该值并不将驱动要加上的数据帧头或 PPP HDLC 帧头计算在内。

最大的以太网数据报为 1518 个字节，其中报头占了 14 个字节，校验码又占用了 4 个字节，所以只有 1500 字节的付载，以太网接口的 MTU（最大传输单元）就是 1500，PPPoE 协议又把报头的长度增加了 6 个字节,再加上 PPP 协议标志在数据报中所占的 2 个字节,所以 PPPoE 的 MTU 也只能是 1500-8=1492，故此处应设为 1492。

MaxTransmit 是指示微端口同时可以处理的数据报的个数，NDISWAN 则确保不会让超过这个数目，在系统开始时，我们将它设为 1，可以确保 NDISWAN 每发一个数据报，需要在前一个数据报处理过才继续，这样就简化了微端口的处理。

HeaderPadding 和 TailPadding 是数据报的报头填充字节数和报尾填充字节数，设置为 0。

Endpoints 微端口所支持的最大的点对点连接数目，理论上，PPPoE 在一条线路上所支持的连接数只受系统资源的限制，但是我们为了安全起见，还是给它设置为 10。

MemoryFlags，HighestAcceptableAddress 是有关硬件访问内存的参数，由于我们的微端口不直接同硬件打交道，所以没有 DMA 的属性，故可以忽略。

FrameBits 指定驱动所支持的数据帧格式，这里设置为 PPP_FRAMING 同 TAPI_PROVIDER 的组合。前者表明我们的微端口接收 PPP 封装的数据，后者则是告诉 NDISWAN，该微端口支持 TAPI，可以对 TAPI OIDs 进行响应。

以上的结构在以后的 PPP 握手过程中有可能会改变，NDIS 同样采取类似的方法，调用 MiniportSetInformation 来改变。

在一系列的 OID 得到正确响应后系统才会最终承认该微端口为 ISDN 卡。

这些 OID 包括：

OID_GEN_MAC_OPTIONS

OID_GEN_MAXIMUM_SEND_PACKETS

OID_WAN_CURRENT_ADDRESS

OID_PNP_CAPABILITIES

OID_GEN_SUPPORTED_LIST

OID_GEN_SUPPORTED_GUIDS

OID_WAN_MEDIUM_SUBTYPE

OID_WAN_GET_INFO

4.3 TAPI 支持

Windows 组件 NDISTAPI 在 NDISWAN 之上，向上对 TAPI 用户程序提供服务，向下又同 NDISWAN 接口完成系统内核态的操作。

本节简要描述 WAN 小端口驱动是如何利用 OID 进行唤醒线路，发起呼叫，建立连接以及关闭呼叫等操作的。

WAN 小端口初始化为支持 TAPI 后，应用程序就可以发送电话请求（Telephony Requests）了，通过用户态的 KMDDSP 服务模块，将该请求转换为相应的 OID，再传送到 NDISTAPI 模块，NDISTAPI 该 OID 传递到 NDISWAN，NDISWAN 又继续下传该 OID，导致 NDIS 调用下层小端

口的 MiniportQueryInformation 或 MiniportSetInformation，然后小端口驱动再进行相应处理。

微端口将线路的状态的转变及时地通过调用 NdisIndicateStatus，向上层驱动传递，然后这一状态又层层上传至 TAPI 应用程序，TAPI 应用程序再根据情况执行新的操作。

在系统中，在用户区执行 TAPI 功能的是 explore.exe 所启动的一个客户线程完成的，另外一个应用程序是 rasphone.exe，它可能也是启动同样的线程，因为二者的拨号界面都是一样的，都是传统拨号的界面。

在 OID_WAN_GET_INFO 之后，系统会继续获得并重新设置该微端口 TAPI 的属性，发送如下的 OID：

OID_TAPI_PROVIDER_INITIALIZE

WAN 微端口完成它作为一个 TAPI 服务者所需的必要的初始化操作，返回本微端口支持的线路的个数。

OID_TAPI_GET_DEV_CAPS

返回指定连接线路对于 TAPI 支持的能力参数。

OID_TAPI_GET_EXTENSION_ID

返回指定连接线路的扩展 ID

OID_TAPI_NEGOTIATE_EXT_VERSION

返回在握手过程中，微端口所希望遵循的 TAPI 的最高扩展版本

OID_TAPI_OPEN

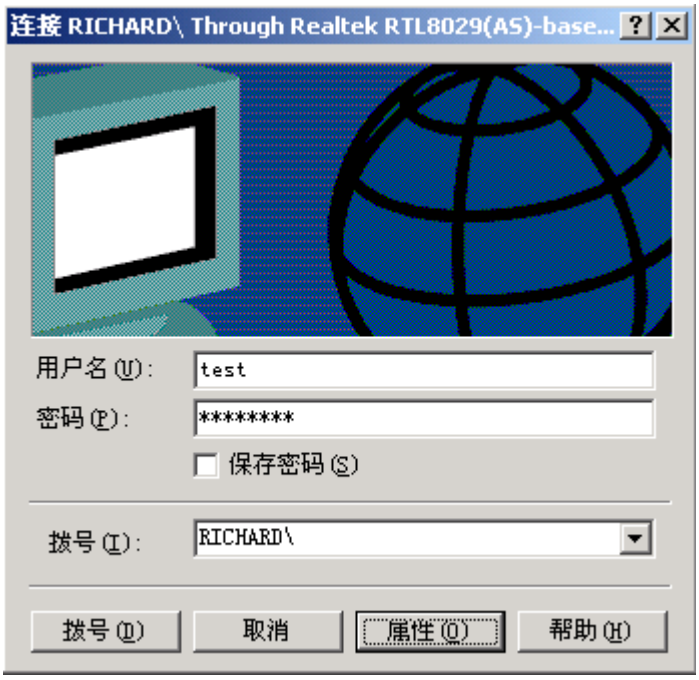
打开指定连接线路的设备，初始化连接线路

OID_TAPI_CLOSE

关闭线路

OID 处理到此处，协议安装过程宣告完全结束，同时中间层驱动的另一部分—协议也应该已经完成了初始化，绑定到网卡，设置网卡属性等操作。

这时用户可以开始拨号了，先用 Utility 建立同服务器 AC 的连接项，添加到电话簿上，然后打开系统的拨号网络，双击该图标，就会出现拨号界面（图表 16）。



图表 16 标准拨号界面

用户点击拨号时，会导致 WAN 微端口收到 OID，OID_TAPI_MAKE_CALL，而作为参数传送下来的还有关于所用线路的信息以及本次呼叫参数，这些呼叫参数就包括用户信息，呼叫目标。

正常的 PPP 连接到这时，就是建立链路层的连接，实现 PPP 认证和配置。在基于 PPPoE 的系统，我们的线路是永久连接的，但是如果没有执行 PPPoE 发现过程，则没有另一端的信息，不能进入 PPPoE 会话阶段而进行 PPP 的认证和其他操作。

所以当此处微端口驱动收到 OID_TAPI_MAKE_CALL 时，正是进行 PPPoE 发现的时机。微端口驱动会调用发现模块，建立 PPPoE 会话，然后向上指示线路的状态为 LINEDEVSTATE_CONNECTED，说明线路已经接通，可以发送 PPP 数据帧了。

然后 NDISWAN 开始同对方（AC）进行 PPPoE 支持下的 PPP 通讯，直到本次呼叫被终止。

4.4 连接控制

PPPoE 同时支持多个会话，为了及时对连接进行诊断并采取相应措施，必须对各连接进行分别控制。

本课题用了两种连接控制方法：

1.PPP 的 LCP 方法

2.PPPoE 方法

4.4.1 利用 PPP 的 LCP 连接控制[23][24][25]

PPP 协议由三个部分组成：

１．封装多协议数据报的方法

２．链路控制协议（LCP，Link Control Protocol）

３．网络控制协议簇（NCPs，Network Control Protocols）

LCP 通过交换数据报来进行建立连接，连接的建立是一系列的选项进行协商的过程，注意这里的选项并不涉及到具体的网络层协议。每次只有一个 LCP 数据报通过 PPP 封装在链路上传递，其格式为：

0 16 24 32 48

|协议|代码|标志符|长度|数据……

协议字段指明所封装的是 LCP 协议，其数值为 16 进制 C021，占用 2 个字节。代码字段指明了 LCP 的种类，最常见的 LCP 代码及代表的意义有：

1 命令设置请求（Configure-Request）
2 命令确认应答（Configure-Ack）
3 命令否认应答（Configure-Nak）
4 命令拒绝应答（Configure-Reject）
5 结束连接请求（Terminate-Request）
6 结束连接确认（Terminate-Ack）
7 代码拒绝 (Code-Reject)
8 协议拒绝 （Protocol-Reject）
9 应答请求 （Echo-Request）
10 应答回应 （Echo-Reply）

标志符字段占用 1 个字节，用来对双方发送的请求和相应的回应进行匹配。长度字段用 2 个

字节来指示包括代码字段、标志符、长度和数据字段的总长度。前 8 种都是关于选项设置的链路控制帧，而 9 和 10 则是用来监测链路的状态，性能测试以及其他功能。任一方接收到应答请求后，都必须返回应答回应；但是这两种数据帧只有在选项配置完成之后才可以发送，否则该帧会被对方丢掉。

一般情况下，当建立 PPP 会话之后，服务器会周期性发送应答请求，来监测是否客户端会异常中断。正常情况下，这个请求/回应的过程是由 NDISWAN 处理的，但是在实际工作状态中，客户端并不向服务器主动发送应答请求，来监测服务器端是否会异常中断连接，这样就会形成一个假连接。在客户端驱动中需要来主动监测这种异常情况，那么如何进行呢？

在代码为 9、10 的两种数据帧中，数据段的头 32 位是一个被称为 Magic-Number 的数值，它用来侦测回环连接或者和其他数据链路层出现的异常。Magic-Number 是不停变化的，其生成算法我们无从知晓，但是要想主动发送应答请求就必须提供一个 Magic-Number。

经过试验，我们发现利用客户端向服务器端发送的应答回应帧，可以来生成自己的应答请求帧。具体做法如下：

截获 NDISWAN 下发的应答回应帧，将其中的代码字段由 0x0a 改为 0x09，然后将其后的数据全部拷贝到内存缓冲区中。经过这样处理的缓冲区就保存了一个固定的应答请求数据帧。虽然其中的 Magic-Number 是固定的，但是经过实际检验，还是可以得到 AC 端的回应的。

有了待发送的数据，就可以建立监测机制了。

有三个部分异步配合完成检测：

1. 驱动发送端。指从 NDISWAN 接收待发送的 PPP 数据，经 PPPoE 打包后向网卡传输的模块，主要完成检测机制的启动。
2. 驱动接收端。指自网卡收到对方发送的数据，解包，再形成 PPP 帧，上传至 NDISWAN 的模块。主要记载是否接收过对方发送的有效数据。

3．定时检测模块。周期性对连接状态进行检测。

主要控制变量及其初值为：

BOOL PPPSessionEstablished=TRUE;

注释：PPP 会话是否已经建立，以客户端是否收到应答请求为标志，如果客户端收到过应答请求，则连接的双方已经建立了 PPP 会话。

BOOL PacketReceivedSinceLastCheck=FALSE;

注释：指示在上一次检测连接之后有没有收到过对方发送过来的数据帧，如果收到过，则表明连接正常。

BOOL EchoRequestSent=FALSE;

注释：记录客户端是否已经发送过应答请求帧

发送端算法：

```
BEGIN
－>接收到待发送数据
if   PPPSessionEstablished
then [监测 PPP 数据]
                    if 是应答回应帧
                    then
```

```
                    PPPSessionEstablished=TRUE; /*设置 PPP 会话建立标志*/
                    PacketReceivedSinceLastCheck=FALSE;
                     EchoRequestSent=FALSE
                    [拷贝该帧到预先分配的缓冲区，并修改代码字段为 0x0a]
                    [初始化定时器]
                endif

    endif
    [PPPoE 打包原来的数据帧并发送]
END
```

接收端算法：

```
BEGIN
－>接收到待上传数据
PacketReceivedSinceLastCheck=TRUE
[上传数据解包后的数据帧到 NDISWAN]
END
```

定时检测模块算法：

```
BEGIN
  IF   !PPPSestionEstablished
  THEN
          return;
  ELSE

  IF PacketReceivedSinceLastCheck
  THEN
            PacketReceivedSinceLastCheck=FALSE;
          EchoRequestSent=FALSE;
          return;
  ENDIF
  //没有收到数据
  IF EchoRequestSent
            //发送过回应请求，而在此期间没有收到过数据
            //说明本次连接已经中断
  THEN
            [向系统报告连接状态为没有相应]
            //以 INEDISCONNECTMODE_NOANSWER 向 NDISTAPI 报告
            //NDISTAPI 继而断掉改次连接，结束会话
  ELSE
             //如果之前并没有发送过则在此处发送回应请求
             [发送回应请求]
          EchoRequestSent=TRUE;
```

```
  ENDIF
END
```

从上述算法可以看出，在接收端并未检测接收到的数据是否应答回应帧。这是正确的，既然可以接收到有效的 PPPoE 数据，则表明此连接仍是活动的，那也就没必要再检测每个接收到的数据了，从而可以提高整体性能。

用这种机制每 5 秒钟检查一次连接状态，这样最多只用两个周期－10 秒内可以及时断掉可能出现的假连接。

4.4.2 PPPoE 连接控制

在 PPPoE 会话阶段，任何时候，任何一方只要收到或者发出 PADT（PPPoE Active Discovery Terminate）数据报，则表示本次会话已经终止，那么就不允许任何的 PPP 数据再通过本会话进行传输，即使是正常的 PPP 终止帧也不允许在 PADT 之后传输。

通常而言，PPP 连接的会话双方应该用 PPP 的终止帧来结束本次会话，但是当 PPP 不可用时，PADT 的控制作用还可以起到同样的作用。

在驱动中，我们只在接收模块中对每个数据帧进行扫描，一旦发现该帧为 PADT，且对本次会话有效，那么就终止本次会话，并释放资源。

4.5 PPPoE 支持

PPPoE 协议的详细描述在 RFC2516，通过第二章的介绍，知道 PPPoE 会话的实质就是在以太网上的 PPP 会话。PPPoE 支持虽然是整个课题的核心，但是并不复杂。本节从函数功能和数据结构两方面对 PPPoE 支持做一介绍。

4.5.1 数据结构

PPPoE 的控制，主要集中体现在 PPPoE 发现阶段。RFC2516 约定，在此阶段，PPPoE 数据帧的负载（payload）包含 0 个或多个标签（tag）。每个 tag 都是 TLV（type-length-value）结构，tag 的定义如下：

```
typedef struct _PPPOETAG {
        USHORT type:16;  /* 2 个字节，标签类型  */
        USHORT length:16;      /* 2 个字节，标签长度，标签内容的长度  */
        UCHAR tagpayload[1500]; /*  标签内容  */
}PPPOETAG;
```

因为一个以太网数据报最大传输单元（MTU）是 1500，所以定义 tagpayload 为占用 1500 个字节的字符，才不至于在偶然情况下发生溢出。

PPPoE 目前定义了 9 个标签，分别是：

0x0000 结束符（End-Of-List）。该标签表明数据报的负载到此为止，此标签的长度恒为 0。

0x0101 服务名称（Service-Name）。表明客户端感兴趣的服务或 AC 服务器所提供的服务。此类标签的内容是 UTF8 字符串，当标签长度为 0 时，意味着任何一种服务都是可接受的。

0x0102 AC 名称（Access Concentrator-Name）。访问集中器也即是服务器，该标签表明服

务器的名称。

0x0103 客户机单一标志（Host-Uniq）。该标签是客户机用来将 PADI 和 PADO，PADR 和 PADS 之间建立匹配的，标签内容可以是客户机任意定义的二进制值。AC 如果收到含有此类标签的 PADI 或 PADR，不能更改该标签的值，而应该在相应的 PADO 和 PADS 中原封包含该标签，客户机利用这个标签，就可以知道这是对哪个请求的响应。

0x0104 AC-Cookie，AC 有可能在 PADO 中包含这样一个 cookie，用来防范可能存在的 DOS 攻击（详见下文）。客户机收到这样的 PADO 之后，必须原封在后继的 PADR 中包含该标签。

0x0105 提供商信息，包含服务器提供商的一些信息。

0x0110 会话中继 ID（Relay-Session-Id）

0x0202 访问集中器系统错误（AC-System-Error）。该标签说明 AC 出现故障，已不能处理用户的请求，并且在标签内容中包含故障的性质。这种标签有可能出现在 PADS 数据帧中。

0x0203 一般错误（Generic-Error）标签。提示有不可恢复性错误出现，这种标签可能出现在 PADO,PADR,PADS 中。

通过这些标签的综合使用就可以完成 PPPoE 发现，而在驱动实现中，除了必须对这些标签识别，还要有全局的控制量作为 PPPoE 操作的依据，下面是这个数据结构的定义：

```
typedef struct _DiscoverElement
{
    USHORT SESSIONID;*
   //PPPoE 会话 ID
    TAGTAG server[MAX_ACNUMBER];
   //服务数组，用来保存发现的服务
    UCHAR servermac[MAX_ACNUMBER][6];
   //用来保存 AC 的网卡物理地址。
    TAGTAG service[MAX_ACNUMBER];
   //保存服务列表
    TAGTAG cookie[MAX_ACNUMBER];
   //保存 AC 发过来的 cookie
    int ACnumbers;
   //AC 的数量
    int ACChoice;
   //所选择 AC 的索引号
    UCHAR localmac[6];
   //本地 MAC 地址
    UCHAR peermac[6];*
   //对方的 MAC 地址
    TAGTAG dialserver;
   //从拨号应用程序中传递过来的服务器名称
    TAGTAG dialservice;
   //从拨号应用程序中传递过来的服务名称
}DiscoverElement,*PDiscoverElement;

typedef struct _TAGTAG{
    USHORT length:16;
```

```
    UCHAR tagpayload[1500];
}TAGTAG;
```

DiscoverElement 的值是在不同的过程中逐步得到填充的，所以该结构必须是全局的。正如第二章所说，PPPoE 发现的最终目的就是得到会话 ID 和对方 MAC 地址。在下一节中将对 PPPoE 发现的具体实现做一描述。

4.5.2 PPPoE 发现的实现

执行 PPPoE 发现时，涉及到数据的发送，接收，分析，存贮等。为了将注意力集中在 PPPoE 协议本身，本文把发送和接收放在 4.6 做较为详细的介绍。

4.5.2.1 Discover 过程

PPPoE 发现由过程 Discover 完成它的原型为

```
BOOLEAN Discover(PLINK Link,const PUCHAR pDialString, ULONG len)
```

在 4.3 中，本文指出，进行 PPPoE 发现的时机是在微端口驱动收到 OID_TAPI_MAKE_CALL 时。微端口驱动会调用 Discover，建立 PPPoE 会话，这一过程相当于取代了 PPP 协议中的线路连接。

等待 AC 数据时，需要引入超时机制，还要有容错机制。

容错是这样实现的：在发送 PADI 后如果过了一段时间还没有收到 PADO，则重新发送一次 PADI，并延长等待时间。这样的反复最多进行三次，如果还是没有得到响应，则表明并无 AC 存在或网络出现故障。同样的容错处理还在发送 PADR 并等待 PADS 时采用。

Link 等参数都由微端口驱动传递过来，通过变量 Link，Discover 可以得到当前网卡的描述结构，及 DiscoverElement 的实例 pElements；通过 pDialString，Discover 分析出服务器和服务，用来同接收到的 PADO 中的相应标签做比较。

以下是 Discover 的算法：

```
BEGIN

    [分配待发送数据帧需占用的内存]
    [调用 PPPoECreatePADI 生成 PADI]
    FOR I=1 TO 3
    NdisResetEvent(&Link->PPPoEEvent);
    [发送 PADI]
    NdisWaitEvent(&Link->PPPoEEvent,(PPPOE_NO_REPLY)*(i+1))
        IF (收到数据报)break；
    ENDFOR

    IF(超时)return FALSE;

    [调用 MakeDialTags 生成拨号的标签（dialserver，dialservice）]
    pElements->ACnumbers=0;    //reset the number of AC
    pElements->ACChoice=0;      //reset the choice of user
```

```
FOR 所接收到的每个数据报
    IF(是 PADO)
    THEN
      [判断是否同拨号标签匹配]
        IF(匹配)
        THEN
              pElements->ACChoice=pElements->ACnumbers
              pElements->ACnumbers++;
              Break；
        ELSE
              pElements->Acnumbers++
        ENDIF
    ENDIF
ENDFOR

IF(0==pElements->Acnumbers)return FALSE;
/*没有 PADO，返回错误*/

[拷贝 pElements->servermac[pElements->ACChoice]到 pElements->peermac[i]]
/*得到对方的 MAC 地址*/
[调用 PPPoECreatePADR 生成 PADR]

FOR I=1 TO 3
NdisResetEvent(&Link->PPPoEEvent);
[发送 PADI]
NdisWaitEvent(&Link->PPPoEEvent,(PPPOE_NO_REPLY)*(i+1))
   IF (收到数据报)break；
ENDFOR
/*等待 PADO*/

FOR 所接收到的每个数据报
    IF（是正确的 PADS）
    THEN
    [保存 PPPoE 会话 ID 到 pElements->SESSIONID]
    /*得到会话 ID*/
    break；
    ENDIF
ENDFOR

IF(收到 PADS)
THEN
Link->SessionEstablished=TRUE;
```

/*标志本次 PPPoE 会话已经完成*/
[在网卡描述符中添加该线路的信息]
[调用 FillPPPoEHeaders(Link)，设置固定的 PPPoE 会话的报头]
return TRUE;
ELSE
Link->SessionEstablished=FALSE;
/*建立 PPPoE 会话不成功*/
return FALSE;
ENDIF
END

在上述 PPPoE 发现过程中，不仅得到了会话 ID 已经对方 MAC 地址，而且保存了一个固定的报头，用以会话阶段进行数据报格式的转换。

请注意，接收数据的工作还是由协议驱动做的，关于这一处理，请参照 4.6.1。

4.5.2.2 其他 PPPoE 处理模块

以下是本课题中各模块所调用的有关 PPPoE 数据处理的函数及其简短说明：

void PPPoECreatePADI 生成 PADI

void PPPoECreatePADR 生成 PADR

BOOLEAN PPPoEIsPADO 判断数据帧是否 PADO，同时将包含在 PADO 数据包中的各标签分析出来，返回服务标签和 Cookie。

BOOLEAN PPPoEIsPADS 判断数据帧是否 PADS，同时将会话 ID 取出。

BOOLEAN PPPoEIsPADT 判断是否 PADT

BOOLEAN PPPoECreatePADT 生成 PADT

BOOLEAN PPPoEIsPPPoE 判断数据帧是否 PPPoE 的帧

BOOLEAN Discover 进行 PPPoE 发现

BOOLEAN FillPPPoEHeaders 根据 PPPoE 发现阶段得到的会话 ID、对方 MAC 地址，本机 MAC 地址生成固定的 PPPoE 数据帧的报头，便于在会话过程中数据帧的转化。

BOOLEAN ChooseThisOne 判断从 PADR 得到的服务器和服务，是否同用户指定的一致。

BOOLEAN PPPoESendPADT 发送 PADT，主动结束本次连接。

4.5.2.3 PPPoE 关于 DOS 的防范

DoS 是 Denial of Service 的简称，即拒绝服务，造成 DoS 的攻击行为被称为 DoS 攻击，其目的是使计算机或网络无法提供正常的服务。最常见的 DoS 攻击有计算机网络带宽攻击和连通性攻击。带宽攻击指以极大的通信量冲击网络，使得所有可用网络资源都被消耗殆尽，最后导致合法的用户请求就无法通过。连通性攻击指用大量的连接请求冲击计算机，使得所有可用的操作系统资源都被消耗殆尽，最终计算机无法再处理合法用户的请求。

拒绝服务攻击的最大问题是你几乎无法追踪它。攻击者使用计算机向服务器发送数百万的连接申请，使目标计算机过载。每次申请都有随机选择的回转地址，让人们无法追踪到来源地。

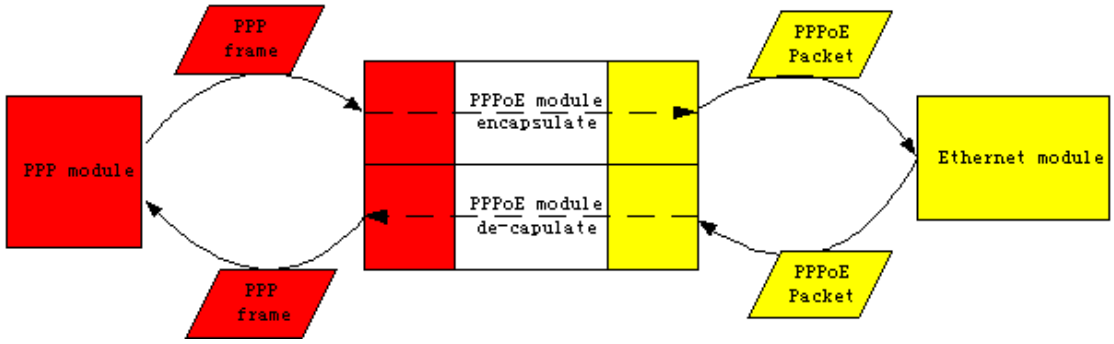
拒绝服务攻击正成为一种更危险更普遍的攻击行为。据加利福尼亚大学研究人员所做的一份报告透露，每周共有 4000 多起类似的攻击事件发生。

PPPoE 协议中关于 AC－Cookie 的设定，可以从一定程度上防范 DOS 的攻击。AC 接收到 PADI 时，会根据数据帧的源地址，采取某种算法生成一个唯一的 AC-Cookie 的值，然后将其包含到 PADO 中并发送给用户机器。按照 PPPoE 协议，如果用户要建立 PPPoE 会话，则客户端必须发送含有 AC-Cookie 的 PADR，AC 收到 PADR 后再次计算出 AC-Cookie，然后同 PADR 中的比较，如果一样则说明客户端的 MAC 地址是可达的，这才为本次会话连接分配资源。

尽管 AC-Cookie 的策略在防范某些 DOS 攻击很有用，但是它并不能消除所有的 DOS 攻击，所以 AC 仍然需要配合其他策略保护自己的资源。

4.5.3 PPPoE 会话阶段

在 PPPoE 会话阶段，发送时只需简单地将 PPP 数据报转化为 PPPoE 的数据报再发送，而接收时只需进行反向转变。数据的转变是比较容易的，困难的是中间层同 NDISWAN，中间层同以太网卡的接口，以及 NDIS 复杂的数据传输机制，这些问题的解决都在别的章节中做了描述。在此只给出数据格式在传输时被转换的示意图（图表 17）。



图表 17 数据格式的转换

4.6 发送/接收机制

4.6.1 接收数据

接收数据由中间层驱动的协议部分负责，协议经正确注册，绑定在机器内的物理网卡之上，并设置网卡的模式后，NDIS 就会将来自网卡的数据作为参数发送到协议所注册的 ProtocolReceive()接口。

假如不事先设置网卡的模式，则协议不会收到任何数据。网卡模式的设置：

广播模式：目的物理地址是 0xFFFFFF 的帧为广播帧，工作在广播模式的网卡接收广播帧。

多播传送（组内广播）：D 类 IP 地址是用于组内广播的，也就是一个人发出的包可以同时被其他多个有资格的人接收，这个人和那些有资格的人就形成了一个组，他们在组内的通信是广播式的。与此相对应，在物理层也存在着组内广播（或多播传送），以多播传送地址作为目的物理地址的帧可以被组内的其它主机同时接收，而组外主机却接收不到。但是，如果将网卡设置为多播传送模式，它可以接收所有的多播传送帧，而不论它是不是组内成员。

直接模式：工作在直接模式下的网卡只接收目的地址是自己的地址的帧。

混杂模式：工作在混杂模式下的网卡接受所有的流过网卡的帧，监控程序就是在这种模式下运行的。

此处我们设定网卡为直接模式，让它只把直接传送到网卡上的以太网数据帧上传至协议。

在协议的 ProtocolReceive（）中，网卡上传来的 PPPoE 数据有两种：

1. PPPoE 发现阶段的数据
2. PPPoE 会话阶段的数据

对于发现阶段接收到的数据，只是用于同 AC 端建立 PPPoE 会话，所以没有必要向上传送。驱动在接收端只需把该数据存入预先分配的缓冲区中就好了，至于处理该数据以及缓冲区的其他操作，则由另外的模块来进行。

PPPoE 会话阶段的数据封装了 PPP 数据，必须向上传送。具体的传送过程是这样的：PPPoE 驱动通过发送过来的 PPPoE 数据帧得到 PPPoE 会话的代码（Session ID），然后得到连接线路的描述指针，再调用 NdisMWanIndicateReceive，将 PPPoE 的纯付载（去掉 PPPoE 帧头的 PPP 数据）传送给 NDISWAN，同时指定数据属于该线路。

4.6.2 发送数据

发送和接收主要由微端口和协议各自分管，但是在异步发送数据时二者又必须通过配合才能实现一个完整的发送过程。异步发送数据是指这种情况：上层协议要发送数据，而下层微端口资源有限却不能立即发送；这时发送模块可以立即返回，进行其他操作。下层微端口在最终将数据发送之后，再通知上层的协议。

在微端口向 NDIS 注册的发送数据函数为 MiniportWanSend，它的函数原型为：

```
NDIS_STATUS    MiniportWanSend(
                                    IN NDIS_HANDLE MacBindingHandle,
                                    IN PLINK      Link,
                                    IN PNDIS_WAN_PACKET Packet
                                   )
```

假如 MiniportWanSend 不能及时发送该数据帧，则必须将该数据帧加入微端口的发送缓冲区中，等待硬件的发送，同时返回一个状态值 NDIS_STATUS_PENDING，说明已将该数据报加入到发送队列中了，而且会在时机成熟时异步发送。在发送结束后应该调用 NdisMWanSendComplete 来通知 NDISWAN，汇报发送情况。

在 Modem 的驱动中，NdisMWanSendComplete 是由硬件中断调用的。但是在我们的这个中间层驱动中，并无对硬件的直接操作，而只是‘假装’了一个硬件（微端口）。而且在实际发送，并非发送的 NDIS_WAN_PACKET，而是重新生成一个 NDIS_PACKET 格式的数据报（当然，这个数据报是遵循 PPPoE 的），然后再调用 NdisSend（）发送出去。调用 NdisSend 时又把这个数据报交给了下层微端口（网卡驱动），下层微端口发送完数据后同样需要 NdisMSendComplete 来通知 NDIS，NDIS 再调用协议的 ProtocolSendComplete 接口，协议再找出原先 NDISWAN 交来的待发送的数据帧的句柄，释放在中间层生成的 NDIS_PACKET ，然后调用 NdisMWanSendComplete 通知 NDISWAN。这样才算最终完成了一次延迟的发送。

所以我们对待发送的数据帧进行从 NDIS_WAN_PACKET 转换成 NDIS_PACKET 的时候，必须保存原先的 NDIS_WAN_PACKET 信息，才能正确调用 NdisMWanSendComplete 来通知 NDISWAN。

通过 PPPoE 封装发送一个 PPP 数据帧的算法可以表示为：

发送端 MiniportWanSend **算法：**

```
BEGIN
/*
NDISWAN 将待发送的 NDIS_WAN_PACKET（p1）传送至微端口的 MiniportWanSend
*/
[依据 PPPoE 协议根据 NDIS_WAN_PACKET 生成一个新的 NDIS_PACKET（p2），同时在 p2 中保存 p1 的句柄]
[调用 NdisSend 发送 p2，并返回值存入 status]
IF status！=NDIS_STATUS_PENDING              //成功或失败
THEN
[释放 p2]
 /*
无论成功或失败，此处都无需调用 NdisMWanSendComplete，对此数据报已经处理完毕，只需返回发送状态即可。
*/
 return status
ELSE
 return status
/*
返回 NDIS_STATUS_PENDING，NDISWAN 等待 NdisMWanSendComplete
*/
ENDIF
END
```

接收端 ProtocolSendComplete 算法

```
BEGIN
/*
以太网卡微端口发送 NDIS_PACKET(pa)之后，将 pa 的句柄传递到协议的 ProtocolSendComplete
*/
[根据 pa 的句柄，找出在数据转换时保存的原始数据帧 NDIS_WAN_PACKET（pb）的句柄]
IF pb！ =NULL
THEN
        [释放 pa]
        [以参数 pb 调用 NdisMWanSendComplete，向 NDISWAN 汇报 pb 发送情况]
ENDIF
    return
END
```

在此函数中，微端口由上层 NDISWAN 处接收 NDIS_WAN_PACKET 格式的数据报，之后重新生成一个 NDIS_PACKET 格式的数据报（当然，这个数据报是遵循 PPPoE 的），然后再调用 NdisSend（）发送出去。

利用 NdisSend 发送数据报，总是不免会因下层微端口的处理延时或硬件的处理能力有限而使得发送暂时不能进行，如果数据报是可以在将来发送的，NdisSend 就会返回一个状态值 NDIS_STATUS_PENDING，说明已将该数据报加入到发送队列中了，而且会在时机成熟时异步发

送。

在下层微端口最终发送了该数据报并向 NDIS 汇报时，Ndis 调用绑定在该微端口上面的协议所注册的 ProtocolSendComplete()，参数包含已发送数据帧的句柄，这时协议驱动查询该数据帧，得到原始 NDIS_WAN_PACKET 数据帧的句柄并向 NDISWAN 汇报发送情况(同样通过 NDIS 汇报)。这才算一个数据帧完整的发送过程。

实际上，NDISWAN 从上层协议接收到的数据报就是 NDIS_PACKET 的，这里就可以看出，数据在发送时进行了反复的转换，这是本文所采取的设计方案的一个缺点，但是，这种中间层驱动以一点效率换来实现上的灵活性在本文这个案例上还是可取的。

第五章 总结与展望

本文在进行 PPPoE 协议实现的研究开发中，主要做了两方面的工作：一是在深入学习研究 PPPoE 协议的基础上提出了在 Windows 系统上实现 PPPoE 协议的方案；二是实现了 PPPoE 协议客户端软件。

PPPoE 客户端软件已经通过实地检测，可以使用户在 Windows98/SE/2000/ME 下完成通过 ADSL 的接入，而且各项数据传输参数都符合 ADSL 传输指标。

由于 PPPoE 协议的提出为时不长，缺乏较为完备的技术资料作为参考，导致了在 PPPoE 客户端软件的研制过程中，从软件系统的总体设计到具体实现，每一步都是艰难的探索求证过程，值得欣慰的是，课题任务最终仍然被较为圆满地完成了。

本文的主要贡献如下：

1. 在国内较早对 PPPoE 协议进行较为全面的研究，本文对各种宽带网络，以及 PPPoE 和其他 ADSL 接入模式所进行的横向和纵向的比较，也有助于 ISP 接入方案的选择。
2. PPPoE 模拟环境的完善。本课题的完成，使得该 PPPoE 模拟环境有了 PPPoE 客户端软件的支持，后续相关的研究开发将可以得到源代码级的数据和资料。
3. 为后继 TAPI 相关的开发提供了翔实的参考资料。在微软提供的资料中，未能详细地说明 Windows 系统将要发送的 TAPI OID 的序列。而本论文则将开发途中经过反复试验后所得到的正确序列记载下来，这样后继的开发者会大为受益，节省开发时间。
4. 可观的经济效益与社会效益。参照 1.5.1 节的保守性估算，如经推广，仅软件费用一项就可以节省国人 3000 多万美元；因服务费用的降低而引起的用户增加，将导致 ADSL 服务商的平均运营成本的降低，这必然是另一项经济效益的体现，而费用的降低必然会提高用户使用宽带上网的积极性。

本文的创新之处在于：

1. 通过适当的软件，经济方便地建立了 PPPoE C/S 模拟环境，为开发研究的顺利进行提供了条件。本文所建立的 C/S 开发环境避开了 AC 和终端用户所需用的硬件，省却了中间不必要的环节，从而能够将精力集中在对 PPPoE 协议及 Windows 系统的研究之上；这种思想从工程的角度来说，具有参考意义。
2. 实现方案的选择。在实用性能允许的限度内，充分利用现有系统资源，减轻设计编码负担。本文结合 Windows 系统的特点，成功地绕过了 PPP 协议的细节，利用 Windows 的拨号网络组件，实现了一个精致实用的 PPPoE 客户端软件。这种方法既节省了研发的工作量，又大大减少了技术支持人员日后进行系统维护的劳动强度。同时，本方案提供给最终用户的界面仍然是用户所熟悉的拨号网络界面，有很强的亲和力，从用户心理学角度来讲跟其他同类产品相比具有相对优势。
3. 本文不仅实现了 PPPoE 协议，而且针对实际运行过程中可能出现的异常现象，为软件增加了相应的功能特性。如连接控制中截获 PPP 的响应请求帧，生成新的请求帧，建立定时检测机制，实现超时控制等等，这一系列的功能都保证了连接故障的及时诊断和正确应对。

本课题由于时间和技术的原因，尚存在一些有待完善的地方， 比如应该做一个安装程序，自动完成对一般用户来说相对较复杂的安装步骤；另外由于条件限制，缺乏 AC 硬件，对服务器端的研究不够，这也是本文的一个缺憾。

在本课题研究和开发的基础之上，如果条件允许，还可以作如下的进一步研究：

1. 增加 AC 服务器模拟功能。如同 RasPPPoE 那样，只要在 TAPI 支持模块中增加对拨入的支持代码，并在 PPPoE 支持模块中增加对 PADI、PADR 的响应（包括 DOS 防御中，AC-cookie 的生成算法等）。
2. 网络监控。在该软件中，已经实现截取网卡上的数据包，如果增加一个协议分析模块以及相应的应用程序，完全可以实现网络监控，使之具有防火墙的功能。

3．利用本课题的研究成果，可以进行一系列的有关 ADSL 传输性能的测试。

目前我国已有至少 14 个城市开通了 ADSL 接入服务，但是由于市场和中国电信的策略的原因，并未能充分地实现 ADSL 的带宽优势，这是令人遗憾的。另外，宽带服务的内容匮乏以及价格过高仍然是令宽带接入商挥之不去的阴影。不过参照国外 ADSL 的发展势头，这种非技术性的局限必然是暂时的，我们有理由相信，国内宽带接入的明天会更美好。

致谢

两年半的学业即将完成，一路走来，再回首时感想颇多，而更多的则是感激。

首先感谢我的导师陶兰教授。您不仅教给我学识，还教会我严谨的治学态度；慈母般的关怀，时刻给我这离家的学子以温暖和慰藉；不遗余力的传授和支持，使得我在短短的时间内，收获甚丰。忘不了您的细心教诲，您眼中的小孩会尽快成长，以奋力拼搏来报答所有一切。

感谢北京创新浩瀚的赵玲小姐，是她同我一起渡过项目最艰难的时期，也是她同我分享成功后的喜悦。她的坚韧与宽容感动并随时影响着我；在我的论文写作期间，她也提出了很多宝贵的建议以及无私的帮助，在此希望她能一生幸福。

感谢同为陶老师学生的兄弟姐妹，我们就是一个大家庭，感谢你们对我的帮助，一起渡过的快乐时光，将是回忆里永恒的驻足地。

最后感谢我的父母和我永远的朋友小许。是他们支持我坚强面对所有困境，他们使得我所有的一切才变得有意义，他们是我生活、奋斗的力量源泉。

参考文献

[1]"宽带接入的应用与发展",《通讯世界》, 2000/08
[2]"xDSL 技术及其应用", 洪培林, 中国科技大学
[3]"ADSL 在宽带接入网中的应用", 曾志达
[4]《高性能网络技术教程》, M. A. Sortak, F. C. Pappas 等著，钟向群，冬青译，清华大学出版社
[5]《全球宽带产业发展透视》通信信息报 张茂州，2001 年 12 月 05 日
[6]《XDSL 技术与体系机构》Padmanand Warrier, Balaji Kumar 著，任天恩译
[7]《接入网技术要求－ADSL.lite，YD/T 1064-2000》，中华人民共和国信息产业部，2000-06-27
[8]"General Introduction to Copper Access Technologies", ADSL Forum, 1998
[9]"Asymmetric Digital Subscriber Line Introduction", ADSL Forum
[10]White Paper, "Architectural Option for Subscriber Access to Broadband Services", Efficient Networks, Inc
[11]White Paper, "PPP over Ethernet A Differentiating Advantage for Internet Service Providers and Carriers", RedBack Inc.
[12]RFC 1483, ”Multiprotocol Encapsulation over ATM Adaptation Layer 5", Juha Heinaren, 1993/07
[13]RFC1577, “ Classical IP and ARP over ATM”, M. Laubach, 1994/01
[14]RFC2364, “PPP Over AAL5”,G. Gross, 1998/07
[15]RFC 2516,"A Method for Transmitting PPP Over Ethernet（PPPoE）", L. Mamakos et al, 1999/02
[16]技术白皮书, "PPPoE 与现存 PC-to-xDSL 技术的比较", RedBack Inc.
[17]《中国互联网络发展状况统计报告》，CNNIC, 2002/1
[18]Proceedings of the 4th Annual Linux Showcase & Conference, Atlanta,"A PPPoE Implementation for Linux", David F. Skoll, Roaring Penguin Software Inc. , 2000/10/10
[19]《全球 ADSL 挑战有线电视》，通信信息报，2001 年 10 月 18 日
[20]"Introduction to TCP/IP", Microsoft Corporation, 1998/09
[21]《Inside Windows 2000 Third Edition》, David A. Solomon, Mark E. Russionovich, Microsoft Press.
[22]《Developing Windows NT Device Drivers》, Edward N. Dekker, Joseph M. Newcomer, ISBN 0-201-69590-1
[23]RFC 1661, "The Point-to-Point Protocol（PPP）", W. Simpson, Editor, 1994/07
[24]RFC 1332, "The PPP Internet Protocol Control Protocol (IPCP)", Merit, 1992/05
[25]硕士论文, "宽带计算机局部网络信道访问控制协议综合研究", 吕光宏, 电子科技大学
[26]"PPPoE and Linux", David F. Skoll, Roaring Penguin Software Inc., 2000/02

附录：PPPoE 的状态转换