摘要

随着网络技术的发展和 INTERNET 的普及，越来越多的信息资源放在了互联网上，广大的中小企业也都逐渐地构建起了自己的局域办公网，这一方面给用户带来了方便，提高了工作效率，另一方面，也使这些企业对计算机网络的依赖性越来越强。面对日益上升的网络犯罪趋势和日益泛滥的网络信息垃圾污染情况，网络安全问题面临着重大的挑战。基于局域网安全和提高管理效率等多方面的考虑，目前对分析、诊断、测试网络性能和安全性工具的需求不断增加，这些工具需要对在网络上流动的数据进行实时捕获并分析，以实现对网络状态的监听，网络监听对网络网络故障的判断和网络管理具有重大的意义。

通过对网络数据包及包捕获的概念和基本方法、WinPcap 的体系结构、包捕获驱动机制、协议分析及过滤等基本原理进行的深入研究，设计并实现了基于 WinPcap 的网络监听系统。所做的主要工作包括：

1. 对 WinPcap 的体系结构、包捕获驱动机制、协议分析及过滤等基本原理进行了深入的研究，并利用 WinPcap 进行网络数据包捕获和过滤的试验研究。

2. 在解决 WinPcap 丢包问题上，提出了使用两个线程的办法，其中一个线程不停的向驱动程序发送读请求，使得 WinPcap 的读队列始终不为空，当 WinPcap 收到数据时，总是有读队列在那里等待满足需求；另一个线程负责检查这些读请求完成的情况，每检测到一个读请求成功完成时，就对数据实施过滤规则，并决定是否通知用户界面线程处理。

3. 设计、开发了面向企业的监听系统，阐述了系统各功能模块详细的设计和实现过程，在企业实地对所设计的监听系统进行了测试，给出了测试结果。

关键词：包捕获，网络监听，网络安全，数据包分析

Abstract

Along with the development to network technologies and popularization of various network applications, more and more information is put onto Internet, Numerous enterprises have built their network for working, as a result, they increase their work efficiency greatly but depend upon it more and more. In the face of the growing trend of computer-related crime and the proliferation of information networks, increasing garbage and pollution, network security has faced major challenges. Considering the network security and promoting work efficiency, the tools for analyzing , diagnosing and testing the performance and security of the network is increasing needed. These tools have to attain data in the line when the network is in use and sniffer the network data in real time. Now, sniffer is playing an important role in the network security.

The work of this paper is based on the deep research to the concept and the method of network data packet and packet capture, the architecture of WinPcap and the driven mechanism of packet capture, and the principle of protocol analysis and filtering. As a result a set of software system about sniffer is designed and implemented. The work mainly include:

1. We do some deep research on the concept and the method of network data packet and packet capture, the architecture of WinPcap and the driven mechanism of packet capture, and the principle of protocol analysis and filtering, and made some experiment's study on the packet capture and filtering based on the WinPcap.

2. In order to solve the problem of WinPcap packet loss, dual-thread model was adopted, one thread sent the read-request to driven program all the time, this made WinPcap read-list never null, when WinPcap received the data, there are read-lists always waiting for satisfy. Another thread is for checking the accomplish of these read requests, when a read request was satisfied successfully, the filtering rules were actualize to the data, then the read-lists will never null.

3. Proposes the procedure of the detail design and implement of the network sniffer system software, include each module of the system. Finally, we test and evaluate the performance of our sniffer syatem on a LAN of a company.

Key words: Data packet capture, Network sniffer, Network security, Data packet analysis

独创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：李X素

日期：2006 年 10 月 24 日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在_____年解密后适用本授权书。

本论文属于

不保密☑。

（请在以上方框内打“√”）

学位论文作者签名：李X素　　指导教师签名：徐兰芳

日期：2006 年 10 月 24 日　　日期：2006 年 10 月 25 日

1 绪 论

1.1 网络数据包与包捕获技术简介

网络数据包是通过网络传输数据的基本单元，它包含一个包头(header)和数据本身，其中包头描述了数据的目的地以及和其它数据之间的关系[1]。通过网络传输的原始数据必须按照一定的大小和标准进行层层分组打包，并加上相应的包头，封装成为一个个独立的数据包后，才能进行传送。网络数据包捕获技术指的就是在网络数据包传输的过程中对其进行拦截和解密。由于现代操作系统已经封装了所有的网络底层操作，自动实现了网络数据包级的传输控制，所以要完成对网络数据包的拦截，必须要通过操作系统或直接嵌入到操作系统中才能实现。通过对捕获的网络数据包进行分析和过滤，可以达到网络窃听、流量监控、恶意网络数据隔离等目的，因此网络数据包捕获技术是构建防火墙、网络监控、入侵检测等网络安全系统的基础。

1.2 计算机网络安全的国内外研究现状

1.2.1 计算机网络安全现状

随着各种新的网络技术的不断出现、应用和发展，计算机网络的应用越来越广泛，其作用也越来越重要。但是由于计算机系统中软硬件和计算机网络本身的脆弱性以及计算机及网络受地理分布的位置、自然环境、自然破坏以及人为因素的影响，不仅增加了信息存储、处理的风险，也给信息传送带来了新的问题。计算机网络安全问题越来越严重，网络破坏所造成的损失越来越大，Internet 的安全已经成为亟待解决的问题。

从目前的情况来看，对计算机网络的入侵、威胁和攻击，基本上可以归纳为以

下几种：①外部人员攻击(75%)；②黑客入侵(17%n)；③信息的泄漏、窃取和破坏；④搭线窃听；⑤线路干扰；⑥拒绝服务或注入非法信息；⑦删除关键信息；⑧身份截取或中继攻击；⑨工作疏忽，造成漏洞；⑩人为的破坏网络设备，造成网络瘫痪。

造成计算机网络的入侵、威胁和攻击的原因主要有以下几个方面[2]:①局域网存在的缺陷和 Internet 的脆弱性；②薄弱的网络认证环节和系统易被监视性；③网络软件的缺陷和 Internet 服务的漏洞；④没有正确的安全策略和安全机制；⑤缺乏先进的网络安全技术、工具、手段和产品；⑥缺乏先进的系统恢复、备份技术和工具；⑦主机的复杂设置和复杂控制；⑧安装了不正确的安全策略；⑨缺乏相应的安全准则、安全规范、安全政策、安全法规，使得无章可循，无法可依；⑩最重要的一条，没有正视黑客、病毒和计算机犯罪所造成的严重后果，舍不得投入必要的人力、物力和财力来加强网络的安全性。

目前，网络安全问题已引起许多国家、尤其是发达国家的高度重视，不惜投入大量的人力、物力和财力来提高计算机网络系统的安全性。近年来，由于我国政府的高度重视，业界有识之士的不懈努力，媒体的广泛宣传和众多黑客事件的相继曝光，国人的网络安全意识有了大幅度的提高。但与世界先进国家相比，我国网络安全的现状仍不容乐观，很多企业的网络安全防护系统仍有待于建立和 健全，尤其是在网络安全产品的研发上,核心技术大多被国外大型 IT 公司所垄断，这对我国政府加强国防安全、改善网络安全环境、提高网络安全技术水平是非常不利的。

1.2.2 网络安全机制及技术措施

目前国内外维护网络安全的机制主要有以下几类[2, 3]：（1）访问控制机制（2）身份鉴别（3）加密机制（4）病毒防护。

针对以上机制的网络安全技术措施主要有：防火墙技术、基于主机的安全措施、加密技术及其它安全措施等技术。

1．防火墙技术

防火墙是近期发展起来的一种安全有效的防范性技术措施，是访问控制机制 、

安全策略和防入侵措施。它是通过在网络边界上建立起来的相应网络安全监测系统来隔离内部和外部网络，以确定哪些内部服务允许外部访问，以及允许哪些外部服务访问内部服务，以阻挡外部网络的入侵。

2．基于主机的安全措施

通常利用主机操作系统提供的访问权限，对主机资源进行保护，这种安全措施往往只局限于主机本身的安全，而不能对整个网络提供安全保证。

3．加密技术

用于网络安全的加密技术通常又有以下两种形式[2, 3]：

（1）面向服务的加密技术　面向服务的加密技术即通常所说的信息加密,它是指利用好的密码算法对某些敏感数据、文件和程序进行加密，并以密文方式存取，以防泄密，其优点在于实现相对简单，不需对网络数据所经过的网络的安全性提出特殊的要求。

（2）面向网络的加密技术　面向网络的加密技术是指通信协议加密，它是在通信过程中对包中的数据进行加密，包括完整性检测、数字签名等，这些安全协议大多采用了诸如 RSA 公钥密码算法、DES 分组密码、MD 系列 Hash 函数以及其它一些序列密码算法来实现信息安全功能，用于防止黑客对信息进行伪造、冒充和篡改，从而保证网络的连通性和可用性不受损害。

加密技术是网络信息最基本、最核心的技术措施，但加密的有效性完全取决于所采用的密码算法，故一般由中央授权部门研制生产，不能自行采用一些密码算法用于网络中，否则后果不堪设想。

4．其它安全措施

包括鉴别技术 、数字签名技术、入侵检测技术、审计监控、防病毒技术、备份和恢复技术等。鉴别技术是指只有经过网络系统授权和登记的合法用户才能进入网络；审计监控是指随时监视用户在网络中的活动，记录用户对敏感的数据资源的访问，以便随时调查和分析是否遭到黑各的攻击，这些都是保障网络安全的重要手段。

1.2.2 目前国内外几种典型的网络安全技术

一个企业级的典型网络安全措施主要包括防火墙、入侵检测、全网的病毒防护、与外部网络的传输加密(VPN 技术)以及网络内部的信息安全控制如网络安全隔离控制卡、指纹引用系统以及服务器上的身份认证机制等。

1．防火墙系统

防火墙系统能增强机构内部网络的安全性，加强网络间的访问控制，防止外部用户非法使用内部网的资源，保护内部网络的设备不被破坏，防止内部网络的敏感数据被窃取。防火墙系统决定了哪些内部服务可以被外界访问、外界的哪些人员可以访问内部的哪些服务、以及哪些外部服务可以被内部人员访问等。要使一个防火墙有效，所有来自和去往因特网的信息都必须经过防火墙，接受防火墙的检查。防火墙必须只允许授权的数据通过，并且防火墙本身也必须能够免于渗透。

从总体上看，防火墙应具有以下五大基本功能[3, 5, 38]：过滤进出网络的数据包、管理进出网络的访问行为、封堵某些禁止的访问行为、记录通过防火墙的信息内容和活动、对网络攻击进行检测和告警。近几年防火墙技术发展迅速，产品众多，而且更新换代快，并不断有新的信息安全技术和软件技术等被应用到防火墙的开发上。国外技术虽然相对领先(比如包过滤、代理服务器、VPN、状态监测、加密技术、身份认证等)，但总的来讲，此方面的技术并不是十分成熟，标准也不健全，实用效果也不是很理想。从 1991 年 6 月 ANS 公司的第一个防火墙产品 ANS Interlock Servic[3, 42]防火墙上市以来，到目前为止，世界上已有很多公司和技术部门在从事防火墙技术的研究和产品开发。国外产品主要有[3, 4]:TIS Firewall Toolkit, Black ICE, FireWall-1, ZoneAlarn, Sidewinder by Secure Computing, F-Secure, CISCO, NORTON 等等，国内防火墙产品主要有：北京天融信公司的"网络卫士"防火墙系统，清华大学及总参第 56 研究所的安全路由器，邮电部数据通信技术研究所的 SJW06 信息代理服务器 PROXY98，上海交通大学的带安全通道的防火墙系统，中科院信息安全技术工程研究中心的 ERCIST 防火墙系统，上海信息港的黑洞式网络防火墙-Net Guard1000、以及天网、瑞友 RSA、瑞星等产品。

2．入侵检测系统

入侵检测是通过监控系统的使用情况，来检测系统用户的越权使用以及系统外部的入侵者利用系统的安全缺陷对系统进行入侵的企图。它根据用户的历史行为，基于用户当前的操作，完成对攻击的决策并记录下攻击证据，为数据恢复与事故处理提供依据。

目前主要有两类入侵检测系统[5]：基于网络的和基于主机的，前者在链接过程中监视特定网段的数据流，查找每一数据包内隐藏的恶意入侵，并对发现的入侵做出及时的响应 ，后者是检查某台主机系统日志中记录的未经授权的可疑行为，并及时做出响应。

3．虚拟专用网 VPN (Virtual Private Networking)技术

VPN 技术可以在远程用户、公司分支机构、商业合作伙伴与公司的内部网之间建立可靠的安全链接，并保护数据的安全传输。与实际的点到点连接电路一样，VPN 系统可被设计成通过 Internet，提供安全的点到点(或端到端)的“隧道”。

一个 VPN 至少提供如下 3 个功能[6]：

（1）数据加密（2）信息认证和身份认证（3）访问权限控制。

根据用户的需求，VPN 可以用多种不同的方法实现。通常情况下，有基于防火墙的 VPN、基于路由器的 VPN、基于服务器的 VPN 和专用的 VPN 设备等。

4．其他一些常规的安全技术和产品

其他一些常规的安全体系如密码技术、数字签名、数字邮戳、数字凭证和认证中心等技术手段可以保护核心秘密并抵御外来非法攻击。这些技术有的在硬件上保证主机和网络的安全，有的在软件上保证主机和网络的安全，如指纹应用技术保证系统登录的安全，安全隔离卡保证主机和网络的物理隔离，以及防病毒技术保证整个网络的信息的病毒防护。

1.3 网络数据包捕获技术与网络安全的关系

必须充分认识网络安全问题的严重性，严格的按照安全规则处理，才可以把

网络安全风险限制在有限范围内。为了增强网络的安全性，人们在 Internet 与局域网之间设置了防火墙，在网络环境中增加了入侵检测系统和网络安全监控系统，而对于普通用户来说最普遍或许也是最方便的则是安装 PC 版的个人网络防火墙系统。以上这些常用的网络安全系统，其底层核心技术之一就是网络数据包捕获技术。网络数据包捕获是构建以上这些网络安全系统的基础，能否高效、稳定、准确、全面的捕获网络数据包已经成为防火墙、入侵检测等网络安全系统能否获得成功的关键。

作为当前最流行的用户平台，Windows 操作系统占据了绝大多数 PC 用户的桌面。如何在源码不公开的 Windows 平台环境下较好的实现网络数据包的捕获，成为在 Windows 环境下实现防火墙、网络嗅探、入侵检测等多种网络安全系统的底层核心技术。长期以来，这一构建网络安全系统的关键性技术一直为国外一些大型计算机公司所垄断，使得该技术在商业上具备较高的敏感性，同时对我国政府提高国防能力、提高网络安全领域产品的竞争力也是非常不利的。

1.4 网络监听概述

网络监听，也称为网络嗅探(sniffer)，sniffer 是利用计算机网络接口捕获目的地为其它计算机的数据包的一种工具，sniffer 的通用意义是网络协议分析仪，在合理的网络中，sniffer 的存在对网络管理人员是十分重要的。系统管理员通过 snifer 可以诊断出大量的不可见的模糊问题，这些问题涉及两台乃至多台计算机之间的异常通信，有些还与各种协议有关。借助于 sniffer，系统管理员可以方便的确定出多少的通信量属于哪个协议，占主要通信协议的主机是哪一台，大多数通信目的地是哪台主机，报文发送占用多少时间，或是相互主机的报文传送间隔时间等。这些信息为管理员判断网络问题和管理网络区域提供了非常宝贵的信息。

使用网络监控技术进行网络故障诊断与分析，可以监听收集到网络中传送的数据，然后对这些数据进行分析以帮助解决在各种多拓扑、多协议网络上的性能问题并排除网络故障，还可以产生报表等数据分析结果，以更好的支持网络的运行。

此外，使用网络监控技术进行完全分析，还可以及时发现各种危害网络安全的

行为，维护网络的安全性，并可通过监控技术实现审计跟踪，这在网络安全问题上具有重要意义。

纵观国内外在网络监听技术中所使用的包捕获机制的方法，大致可归纳为两类：一类是由操作系统内核提供的捕获机制，另一类是由应用软件或系统开发包通过安装包捕获驱动程序提供的捕获机制，该机制主要用于 Win32 平台下的开发。操作系统提供的捕获机制主要有四种[4, 7]：Bpf(Berkeley packet Filter)、DLPI (Data Link Provider Inteface)、NIT(Network Interface Tap)和 Sock Packet。BPF 由基于 BSD 的 Unix 系统内核所实现；DLPI 是 Solaris(和其它 System V Unix)系统的内嵌子系统；NIT 是 Sun OS4 系统的一部分，但在 Solaris/SunOS5 系统中被 DLPI 所取代；Linux 核心则实现了 Sock Packet 的包捕获机制。从性能上看，BPF 比 DLPI 及 NIT 好得多，而 Sock Packet 最弱。SCO Openserver 虽然本身没有内核捕获模 BPF，但有作为可压入内核的 STREAMS 模块 BPF，采用与伯克利 BPF 相一致的概念，但并未提供伯克利 BPF 的所有功能。

Windows 操作系统没有提供内置的包捕获机制。WinPcap(Windows Packet Capture)是 Win32 上的第一个用来捕获数据包的开放系统软件包，它是一种新提出的强有力并且可扩展的框架结构，WinPcap 包含了一系列以前系统所没有的创新特性。

1.5 本文的主要研究工作

本文主要进行了以下的研究工作：

1．对 WinPcap 的体系结构、包捕获驱动机制、协议分析及过滤等基本原理进行了深入的理论研究。

2．对 WINDOWS 环境下基于 WinPcap 的包捕获方法进行试验研究。

3．研究网络监听系统开发的几个主要问题，在此基础上设计并实现了基于 WinPcap 的网络监听系统。

4．测试、评估开发的网络监听系统。

2 Windows 平台网络数据包捕获技术的实现机制

内部网络与 Internet 间的通信，是通过网关转发数据包来实现的，为了实现对网络中被监控主机数据包的控制转发，监控端必须捕获内部网与 Internet 间的通信数据包。本章从 Windows 操作系统架构及 WinPcap 捕包机制出发，分析 Windows 下网络数据包捕获技术的实现机制。

2.1 Windows 平台网络数据包捕获技术简述

Windows 操作系统的总体架构分为两个层次，即应用层（或称用户态）和核心层(或内核态)。应用层是可以直接接触得到的，应用程序（EXE）与动态链接库（DLL）工作在这一层，动态链接库被应用程序调用时就是应用程序的一部分，所以它们并没有本质区别。Windows 核心层下扩展名为 VXD 或 SYS 的程序，称为驱动程序，驱动程序为上层应用程序提供底层支持。

OSI 七层从下至上依次为物理层、数据链路层、网络层、传输层、会话层、表示层及应用层，Windows 操作系统架构与 OSI 七层协议、TCP/IP 协议的关系影射图如图 2.1。

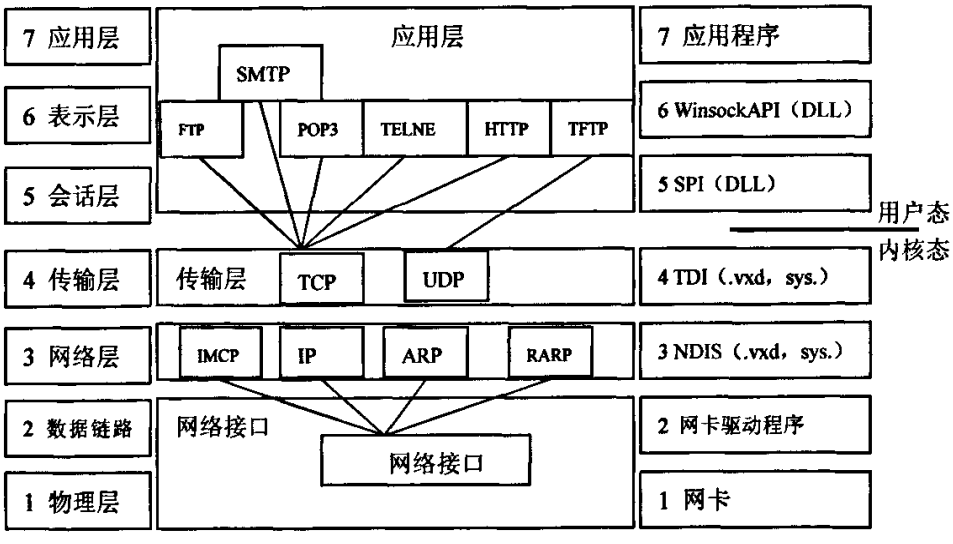

图 2.1 OSI 七层协议、TCP/IP 协议和 WINDOWS 结构的关系影射

从图中可看出 OSI 七层协议在 Windows 体现得非常明显，但因为操作系统并没有严格划分这些层，所以这个关系影射并十分严谨，有时候它们之间会存在一些交叉。

物理层即网卡，其作用一是将线路发来的高频调制信号转化为基带的数据包信号传送给网卡驱动程序，二是将网卡驱动程序传送来的数据包转化成高频调制电流发送到线路上。数据链路层即网卡驱动程序，它负责和 Windows 进行沟通。网络层是 NDIS，NDIS 的重要职能是提供网络层接口，一方面通过网卡驱动程序向网卡传达 Windows 的意思，另一方面，将网卡驱动程序得到的网卡的数据传送给上层 TDI。传输层即 TDI，是传输驱动的接口，负责对信息进行检索、分类并重新组织，TCP 协议的数据包处理就在这层进行。从物理层到传输层都处于核心层，其程序都是驱动程序，表现为.vxd 或.sys。内核态的 Windows 网络体系结构分别见图 2.2。

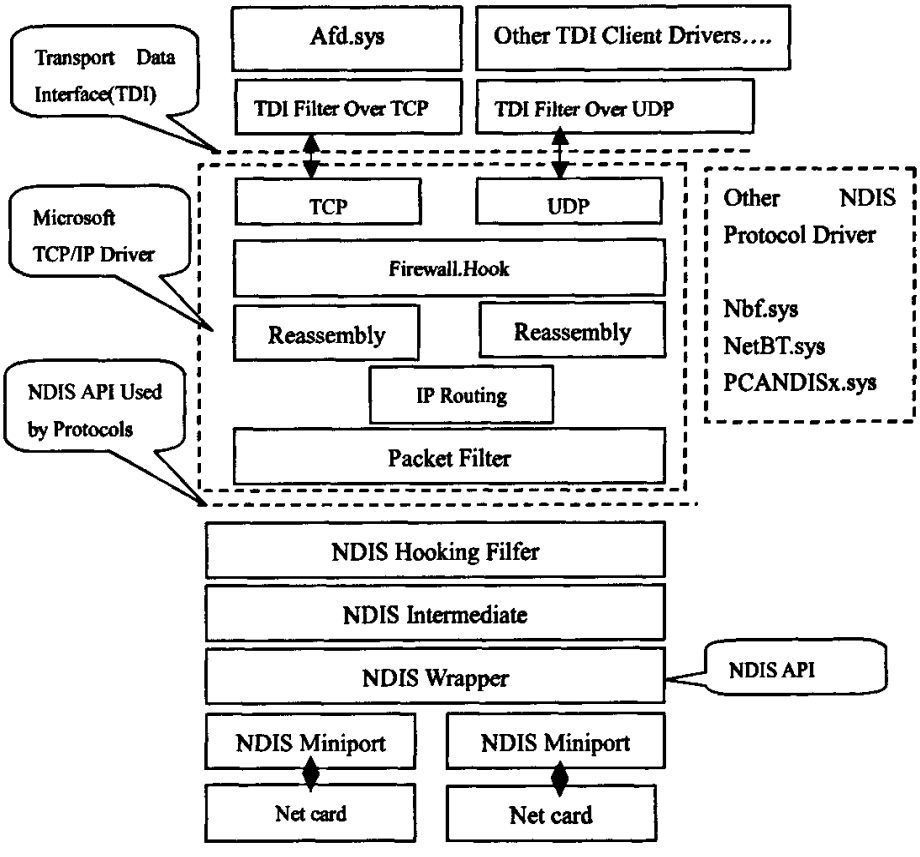

图 2.2 Windows 网络体系结构图———内核态

会话层即 SPI，属于应用层范畴，它的程序为动态链接库（DLL）的形式，负责链接核心层驱程序和高层应用程序，而应用层上最常见的是 EXE 文件，负责数据传输结果显示给用户，并将用户下达的命令传送到下一层。

TCP/IP 的应用层相当于 OSI 的高 3 层，传输层仍相当于 OSI 的传输层，TCP/IP 为网络层换了一个名字称为网络-互连层，是实现 IP 协议的层，TCP/IP 将数据链路层与物理层统称为网络接口层。用户态的 Windows 网络体系结构见图 2.3。

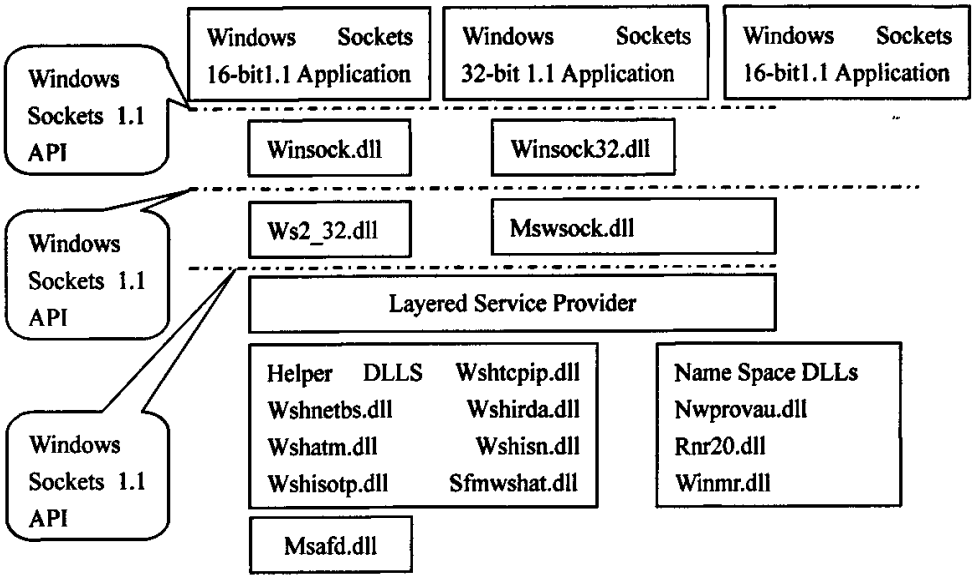

图 2.3 Windows 网络体系结构图---用户态

从这几个图，我们可以看到网络驱动的分层结构，这给我们提供了拦截网络数据包的基本思路。总的来说，要拦截 Windows 下的网络数据包，可以在两个层面进行：用户态（user-mode）和内核态(kernel-mode)。在用户态下有以下 3 种方法进行网络数据包的拦截[4, 8]：

1. Winsock Layered Service Provider(LSP)
2. Windows2000/XP 包过滤接口
3. 替换系统自带的 WINSOCK 动态链接库

显然，在用户态下进行数据包拦截最致命的缺点就是只能在 Winsock 层次上进行，而对于网络协议栈中底层协议的数据包无法进行处理，对于一些木马和病毒来

说很容易避开这个层次的防火墙。

在内核态下网络数据包的捕获方法比较复杂，大多数的个人防火墙都是在内核态下实现的，主要有以下 5 种方法[9-11]：

1．TDI 过滤驱动程序(TDI Filter Driver)

2．NDIS 中 Ifs 层驱动程序(NDIS Intermediate Driver)

3．Win2k Filter-Hook Driver

4．Win2k Firewall-Hook Driver

5．NDIS Hooking Filter Driver

在内核态下能够拦截到比较底层的数据包，但是其实现方法比较复杂，需要编写特定的驱动程序，同时对程序的性能、稳定性等各方面的要求也都比较高。

微软和 3 Com 公司在 1989 年制定了一套开发 Windows 标准[12, 13]，称为 NDIS (Network Driver Interface Specification)，NDIS 提供了大量的操作函数,它为上层的协议驱动提供服务，屏蔽了下层各种网卡的差别。NDIS 向上支持多种网络协议，比如 TCP/IP、NWLink IPX/SPX、NETBEUI 等，向下支持不同厂家生产的多种网卡。NDIS 还支持多种工作模式，支持多处理器，提供一个完备的 NDIS 库(Library)。但库中所提供的各个函数都是工作在核心模式下的，用户不宜直接操作，这就需要寻找另外的接口。

NDIS 为网络驱动的开发提供了一套标准的接口，使得网络驱动程序的跨平台性更好，NDIS 提供以下几个层次的接口[14]：

1．NDIS 小端口驱动(Miniport driver) ，即我们常说的网卡驱动

2．NDIS 协议驱动(Protocol driver)，例如 TCP/IP 协议驱动

3．NDIS 中间层驱动(Intermediate driver)，这是基于链路层和 IP 层之间的驱动

2.2 WinPcap 概述

2.2.1 WinPcap 概述

WinPcap 是由微软资助的一个项目，其核心仍是基于 NDIS 的，但它对 NDIS 进

行封装，是 WINDOWS 平台下一个免费、公共的网络访问系统，它为 WIN32 应用程序提供访问网络底层的能力。WinPcap 是 BPF 模型和 Libpcap 函数库在 Windows 平台下网络数据包捕获和网络状态分析的一种体系结构，如图 2.4 为 WinPcap 简单的结构图。

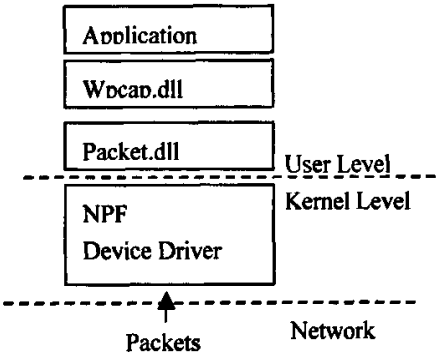

图 2.4 WinPcap 结构图

WinPcap 的优势是提供了一套标准的捕包接口，并与 libpcap 兼容，可使得原来许多 UNIX 平台下的网络分析工具快速移植过来便于开发各种网络分析工具。WinPcap 充分考虑了各种性能和效率的优化，包括对于 NPF 内核层次上的过滤器的支持，对内核态统计模式的支持，WinPcap 还提供了发送数据包的能力。

WinPcap具体功能如下[15, 16]：

1．捕获原始数据包，包括在共享网络上各主机发送/接收以及相互之间交换的数据包 。

2. 在数据包发往应用程序之前，按照自定义的规则将某些特殊的数据包过滤掉。

3．在网络上发送原始的数据包。

4．收集网络通信过程中的统计信息。

2.2.2 WinPcap 的体系结构分析

WinPcap 更详细的体系结构如图2.5所示[15]，由该图可以看出，WinPcap采用的是分层化的驱动程序模型，包含有三个组件：（1）内核级的数据包捕获驱动程序(Npf.sys/Npf.vxd)；（2）低级动态链接库(Packet.dll)；（3）高级系统无关库(Wpcap.dll)。

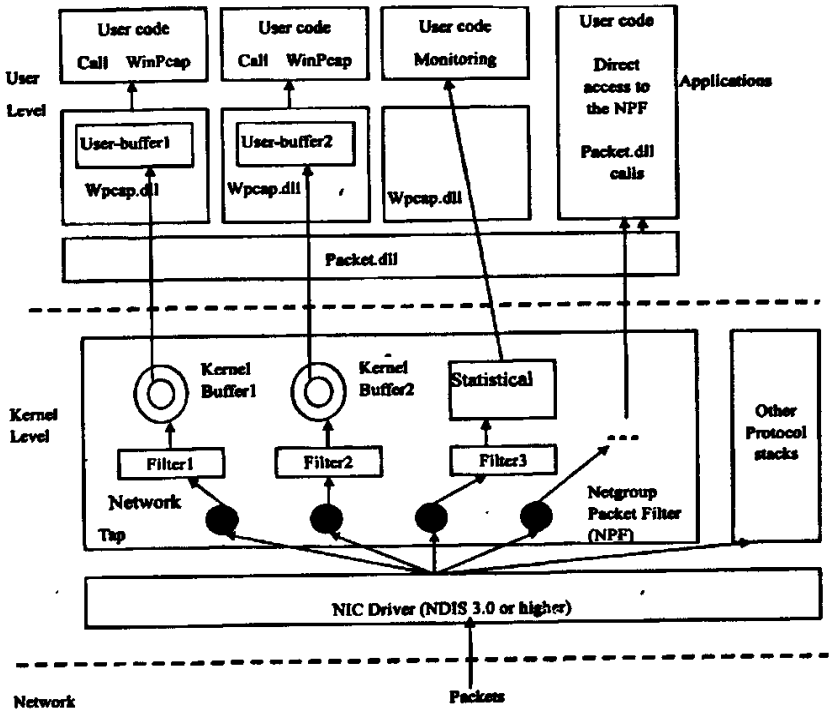

图2.5 WinPcap的体系结构

1．数据包捕获驱动程序(Npf.sys/Npf.vxd)：数据包捕获驱动程序可把设备驱动添加到Windows95、Windows98、windowsNT、Windows2000或者WindowsXP上，它直接从数据链路层读取网络数据包并不加修改的传递给运行在用户层的应用程序，也允许用户发送原始数据包。数据包捕获驱动程序还支持BPF过滤机制，可以灵活的设置过滤规则，并在不同的Windows系统下以不同的文件形式存在，其中在WindowsNT/2000/XP下是Npf.sys，而在Windows95/98/me下是Npf.vxd。

2．低级动态链接库(Packet.d11)：Packet.dll运行在用户层，它把应用程序和数据包的设备驱动程序隔离开来，使得应用程序可以不加修改地在不同的Windows系统下运行。通过packet.dll提供的功能来直接访问BPF驱动程序的包驱动，利用“raw”模式来发送和接收包。不同Windows系统上的packet. dll虽不相同，但它们提供了一套相同的调用接口，即高级系统无关库，它隐藏了用户程序和操作系统交互的细节并完成了下列几个工作：（1）为用户程序提供一套功能强大的抽象接口；（2）根

据用户要求生成过滤指令；（3）管理用户区；（4）负责用户程序和内核的交互。

3．高级系统无关库(Wpcap.dll)：Wpcap.dll是上层的动态链接库，与操作系统无关，和应用程序编译在一起，它含有诸如产生过滤器、用户级缓冲以及包注入等高级功能。它使用低级动态链接库提供的服务，向应用程序提供完善的监听接口。编程人员既可以使用包含在packet.dll中的低级函数直接进入内核级调用，也可以使用由wpcap.dll提供的高级函数调用，这样功能更强，使用也更为方便。

2.2.3 WinPcap 的包捕获驱动机制

WinPcap的包捕获原理来源于UNIX下的带有BPF（BSD Packet Filter）的TcpDutnp。

BPF含有网路开关和包过滤器两个关键组件：网路开关(Network Tap)和包过滤器(Packet Filter)。其中，网路开关从网络设备驱动程序中搜集数据拷贝并转发给过滤器，包过滤器(Packet Filter) 决定是否接收该数据包，以及接收该复制数据包的哪些部分。BPF结构如图2.6所示。

BPF采用有向无圈控制流图CFG(Directed Acyclic Control Flow Graph)模型[16,]，CFG模型解释数据包的解释状态和路径是可记忆的，不需要重复计算。过滤器由应用程序创建，通过Ioctl调用传递给BPF。缓存则由BPF静态创建，大小通常为4K。

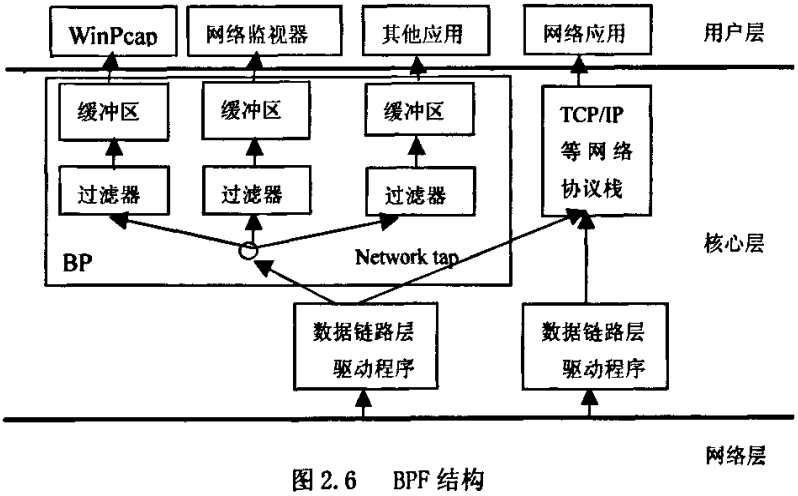

图 2.6　BPF 结构

核心缓冲区分为Store bufer和Hold bufer两部分，其中Store bufer用来接收网卡传来的数据，Hold bufer用来将包从核心缓冲区复制到用户缓冲区。当Store bufer为满而Hold bufer为空时，BPF交换这两个缓存，这样的缓冲区设计使用户程序不需与网卡驱动程序交互，当用户读Hold bufer时，网卡驱动写的是Store bufer。用户缓冲区用来存储从核心区发过来的包，并且由于在用户级别上，它可以防止应用程序直接访问由核心管理的内存。当一个数据包到来时，链路层驱动程序通常将其传送给系统协议堆栈进行处理，但当BPF也在该网络端口监听时，驱动程序将首先调用BPF的Network Tap函数，该函数将数据包传递给每一个应用程序的过滤器，用户程序将决定一个数据包是否被接收以及数据包中的哪些内容应该被保存起来。其中，当过滤器过滤数据包时，数据包仍在链路层的缓存中，并没有拷贝它。当Store bufer满而Hold bufer又不能发生效用时(Hold bufer正在向外读数据)，数据包将被丢弃。

如果过滤程序在用户层运行，则所有数据包必须从核心层拷贝到应用程序然后再在用户层过滤，这样就浪费了系统的CPU时间与内存。而BPF的优点是避免了这种资源的浪费。在多任务环境中，捕包应用程序必须与其它程序共享处理时间，当某个包到来时，捕包程序没有执行，或可能这个应用程序去执行其它任务，此时如果没有核心层的缓存，就会造成丢包，BPF的循环双缓存就是为了避免丢包而设计的。

2.2.4 WinPcap 包捕获机制分析

在Windows NT下WinPcap包捕获驱动和网卡设备驱动的交互是通过NDIS(Network Device Interface Specification)来实现的。NDIS是Microsoft和3COM公司联合制定的网络驱动规范，并提供了大量的操作函数，各个函数都是工作在核心模式的。它为上层的协议驱动提供服务，屏蔽了下层各种网卡的差别，NDIS支持如下三种类型的网络驱动[18, 19]：

1．NIC驱动。NIC驱动直接管理网络接口卡，NIC低端接口与硬件直接打交道，高端的接口允许上层应用向网络发送数据包。NDIS驱动仅能与NDIS中介驱动或协议

驱动通信，而不能与用户模式的应用程序相互通信。NIC驱动可以是小端口驱动或者传统的完整NIC驱动。小端口操作并不直接调用系统例程，它们对于操作系统的接口是NDIS。小端口驱动并不进行数据绑定,它只是将数据传递给NDIS，由NDIS将这些数据上传。

2．中间驱动。中间在上层驱动(传统传输驱动)与小端口驱动之间，与小端口驱动不同，中间驱动看起来象一个协议驱动，一个中间协议驱动可以处于其它中介驱动之上，虽然这样的分层可能对系统性能带来负面的影响。一个包捕获驱动用中间驱动进行开发，可以在传统的传输驱动与小端口驱动之间进行媒介转换，这对于传输驱动来说新的媒介类型可以对NIC进行管理。中间驱动仅与NDIS驱动通信，而与用户模式的应用程序没有关系。

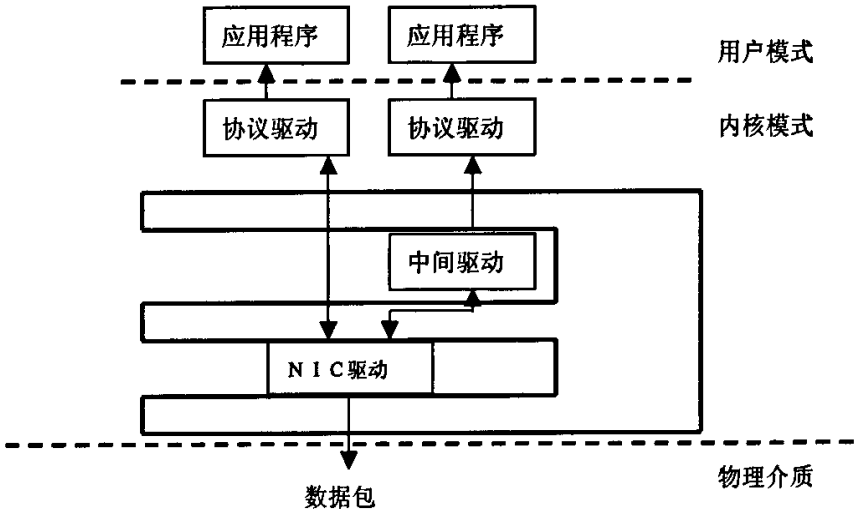

图 2.7　NDIS 结构图

3．协议驱动。协议驱动(传输驱动)通过一个或更多网卡运行网络协议堆栈服务，传输驱动在它的高端为应用层客户提供服务，在低端为一个或更多的NIC驱动与中间驱动服务。

图 2.7为含有两个捕获操作的NDIS结构图，一个与NIC驱动和协议驱动相连，另一个与NIC驱动，中间驱动和协议驱动相连。

2.2.5 包捕获驱动在 NDIS 中所处位置与运作特性

包捕获驱动既与网络驱动通信又与用户应用程序通信， 所以它在NDIS结构中如同一个协议驱动，如图2.8所示[20, 21]。

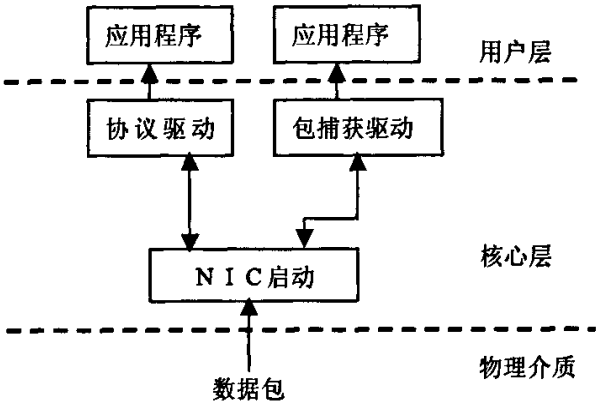

图 2.8 包捕获驱动在 NDIS 中所处位置

包捕获驱动能够独立于硬件运作，在WAN链接中，通常在NDIS上加以虚假的Ethernet头部，这种做法允许协议驱动对WAN链接进行读写操作而不必有任何的改变，但是这却意味着NCP-LCP的数据包将不能被捕获到。协议驱动使用NDIS提供的函数与NDIS低端通信，例如协议驱动必须调用NdisSend或NdisSend Packet才能向NDIS的低端发送数据。

另一方面，低端的驱动与协议驱动以同步方式通信。当一个新的数据包到达时，它们调用协议驱动的回调函数给预先准备好的缓存分配与接收的数据包相对应的缓存大小的指针，在包捕获驱动中这个回调函数是Packet tap。包捕获驱动的运作实际上与标准的协议驱动是完全不同的，表现为如下两点[22]：

1. 包捕获驱动接收当前在网络中传输的所有数据包，可以将网卡设置为混杂模式，协议驱动只能管理广播数据与指定要传送给它的数据。

2. 包捕获驱动不执行一个协议封装与解析过程，但是它存储这些包，给这些数据包加以时间戳，长度等信息并原样不动的传送给应用程序。标准的协议驱动去除

数据包头部数据，将其内部数据传送给应用程序。

在UNIX操作系统中，BPF直接由网络接口的驱动在协议驱动之前调用。对于包捕获驱动来说没有必要，因为它是被看作协议驱动之一，但是它不对硬件层工作，而是在NDIS顶层工作，在NDIS中有优先级。

2.3 利用 WinPcap 进行网络数据包捕获和过滤的实验

在 windows 情况下捕获数据包的结构如图 2.9

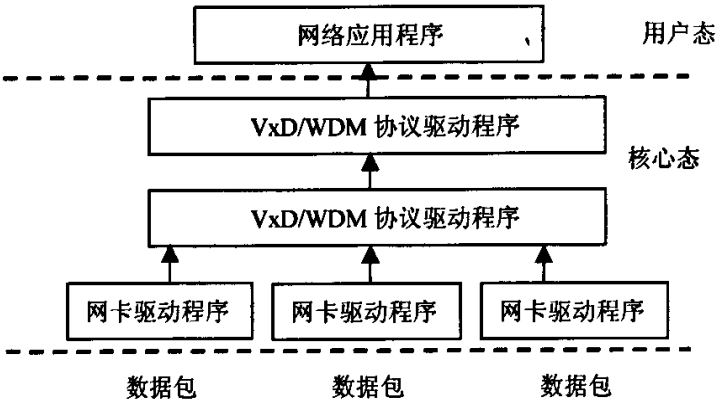

图 2.9 在 windows 情况下捕获数据包

主要实验步骤如下：

1. 将网卡设为混杂模式。

2. 回调函数 Network Tap 在得到监听命令后，从网络设备驱动程序处收集数据包把监听到的数据包负责传送给过滤程序。

3. 当 Packet filter 监听到有数据包到达时，NDIS 中间驱动程序首先调用分组驱动程序，该程序将数据传递给每一个参与进程的分组过滤程序。

4. 然后由 Packet filter 过滤程序决定哪些数据包应该丢弃，哪些数据包应该接收，是否需要将接收到的数据拷贝到相应的应用程序。

5. 通过分组过滤器后，将数据未过滤掉的数据包提交给核心缓冲区，然后等待系统缓冲区满后，再将数据包拷贝到用户缓冲区，监听程序可以直接从用户缓冲

区中读取捕获的数据包。

需要说明的是，如果在一个繁忙的网络上进行捕获，而不设置任何过滤，那得到的数据包是非常多的，可能在一秒钟内得到上千的数据包。如果应用程序不进行必要的性能优化，那么将会大量的丢失数据包。所以，对捕包性能的优化必不可少，可以考虑采用多线程来处理数据包[22]。若在程序中建立一个公共的数据包缓冲池，这个缓冲池是一个 LILO 的队列。于是在程序中使用三个线程进行操作：一个线程只进行捕获操作，它将从驱动程序获得的数据包添加到数据包队列的头部；另一个线程只进行过滤操作，它检查新到的队尾的数据包，检查其是否满足过滤条件，如果不满足则将其删除出队列；最后一个线程进行数据包处理操作，如根据接收的数据包发送新数据包这样的工作都由它来进行。考虑尽可能少丢失数据包的条件，应该是进行捕获操作的线程的优先级最高，这样就可以得到更高的捕包性能。

试验主要源代码及说明见附录 A。

实验程序运行后，首先从适配器列表中选择一个适配器，然后将适配器设置为混杂模式，设置 BPF 包过滤的过滤器参数，分配缓冲池后捕获数据包，把符合过滤器规则的数据转发给网络协议模块分析，最后结束接收，释放数据包对象，关闭网卡。

2.4 WinPcap 常用的数据结构

WinPcap库中几个常用的数据结构[15, 23, 24]简单介绍如下：

1. pcap_t

该结构是WinPcap库的核心数据结构，基本上在程序的开始都要调用pcap_open_live()，函数返回指向这个结构的指针，以后几乎调用库里的每一个函数都要用这个结构作为参数。

2. struct pcap_stat

该结构包含两个可用成员:u_int ps_recv与 u_int ps_ drop 。

3. struct bpf_hdr和struct pcap_pkthdr

内核过滤器每输出一个包，将在输出的数前添加字节的数据，这就是struct bpf_hdr，与之极为相似的是struct pcap_pkthdr，在每一个被捕获的数据包存储到文件中时，这个结构被加在包头。

4. struct bpf_program

此结构如果使用tcpdump表达式，则不用关心其内部结构，但如果要手工编写BPF程序，就需要知道其成员。

WinPcap更详细的数据结构说明见附录B。

2.5 本章小结

本章从Windows操作系统的构架出发，介绍了网络数据包捕获的基本技术概况，并对WinPcap的基本功能、WinPcap的体系结构和包捕获驱动机制、WinPcap包捕获驱动在NDIS中所处位置与运作特性做了详细的分析和论述。在此基础上，设计和进行了利用WinPcap进行网络数据包捕获和过滤的实验，给出了实验的基本方法和步骤，通过实验指出了对捕包应用程序进行性能优化的必要性。本章的研究工作为采用WinPcap作为捕包驱动设计高效的网络监听程序奠定了理论基础。

3 网络监听系统设计的有关问题

本章给出网络监听系统的一般模型，对系统实现过程中几个关键问题进行了研究，详细分析系统的实时性与稳定性，研究数据包过滤条件的选择方法，并提出提高协议分析效率注意的几个问题。

3.1 监听系统的一般模型

监听系统（sniffer）主要用来分析网络的信息流量等信息，以便找出所关心的网络中潜在的问题。网络监听系统的一般模型如图3.1所示[25]。

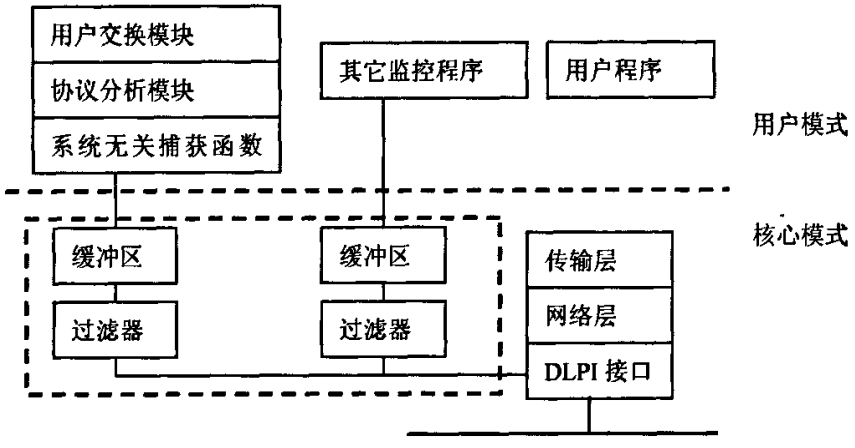

图3.1 网络监听系统的模型

网络监听系统主要由内核空间和用户空间两部分组成[26, 41]。内核空间部分负责从网络中捕获及过滤数据包，用户空间部分负责处理用户界面、协议分析等。如果核心层没有进行数据包过滤工作，还必须负责数据包过滤工作。

网络监听系统一般由硬件、包捕获驱动程序、过滤器、缓冲器、协议分析模块等几种主要组件[27]。

硬件：大部分的sniffer都是在标准的网络适配器之上工作，而不需要特别的硬件，但某些产品可能需要特定的硬件与之配合。在后一种情况下，sniffer甚至可以

分析硬件错误，例如CRC错误，电平问题，电缆问题，网络抖动，帧前导码等。

包捕获驱动程序：这是最为重要的组件，它直接控制网络硬件（如网卡或特殊硬件）从网络上获得数据，并把数据存入缓冲器。

过滤器：过滤器只将用户关心的敏感数据向上层提交，以提高系统的工作效率。

缓冲器：缓冲器用来存放捕获的数据。由于缓冲器的容量有限，它的工作方式通常有缓冲器装满就停止捕获与缓冲器满了还继续捕获两种，但是新的数据会覆盖原有的数据。

协议分析模块：通过对网络数据包的实时或事后的解码分析，可以了解网络上运行的协议和服务，数据帧的源地址和目的地址，数据的帧格式等。

sniffer还包括实时分析统计组件（在捕获的同时找出网络中的问题和故障），数据包编辑和发送组件（对捕获的数据包进行编辑并发回到网络上）。

3.2 实时性与稳定性考虑

企业局域网每天的信息流量很大，为了实现监控的实时性和保障系统的稳定性，比如能实时报警以及不发生大量的丢包情况，这就对捕包模块的工作效率提出了较高的要求。

在捕包模块的设计中，为了能对用户的非法性访问进行实时报警，同时提高协议分析模块分析的效率，在进行系统设计时可考虑在捕包模块中加入部分协议分析功能[28, 30]，这主要是为了检查数据包是否匹配审计信息，并提供实时监控。如果捕获点不加入协议的分析功能，而是将敏感信息的分析完全交给协议分析模块来完成，那么由于协议分析模块要从大量数据包中筛选敏感数据包，这将产生较长延迟时间，从而降低实时性。

此外，还可采用分解 (Slice)技术[29]，也即“只捕获一个包的头n个字节”。采用这种方式的原因是包的最关键部分即协议头都在包的首部，通常建议捕获包头部的至少128个字节的数据以保证能完全捕获到协议头。捕包模块采用Slice技术，可以

增加缓冲区中存放的包的数目，经过优化过滤器的过滤规则可以减少捕包模块拷贝到缓存的数据包的数目，从而降低系统的负载，提高系统的稳定性。

3.3 系统性能的提高

Network Tap和filter都工作在核心层，并且它们的主要功能就是进行数据包的捕获和过滤，所以它们的工作速度是相当快的，一般不会发生数据包的丢失[30, 31]。但是由于捕包模块在监听用户进程和其它进程时是共享CPU的，故当其它用户进程开得比较多的时候，监听进程的工作效率便会大受影响。这时如果网络流量很大的话，就有可能会发生数据包的丢失。为解决这个问题，需要一个足够大的核心缓冲区，来暂时存储符合过滤规则的数据包。当核心缓冲区中的数据包的数量达到一定字节数或是等待一定的时间后，再通过系统调用将核心缓冲区中的数据包拷贝到用户缓冲区中，影响系统性能有以下几个因素[32, 41]：

1．核心缓冲区的大小。如果核心缓冲区设置得过小，那么捕获性能将降低，还有发生丢包的可能，所以通过调用NPF中的核心缓冲区设置函数将缓冲区设置成一个适当的大小，可以明显地提高系统的性能。

2．合理的读操作等待时间。如果核心缓冲区频繁地将采集的数据包送往用户缓冲区，势必加大系统调用所带来的消耗。所以通过设置一个合理的读操作等待时间，来避免数据包刚到达就从核心缓冲区复制到用户缓冲区，而是等待一段时间以使更多的包到达并将之一次从核心复制到用户区，从而提高系统性能。

3．用户缓冲区的大小。大的用户缓冲区可以减少系统调用的次数并因此提高捕获速度。为了获得最好的性能，可以将用户缓冲区设置成和驱动的核心缓冲区相同的大小，这可以避免驱动每次拷贝包到用户缓冲区时都要首先扫描循环核心缓冲区并计算拷贝的字节数，从而提高了拷贝的速度。

3.4 过滤规则

理论上，网络监听系统可以捕获所有网络上的信息，但在实际应用中，有很多

信息在网络管理上并没有意义，如果也捕获并分析这些数据，就会严重影响监听系统的工作效率。可以通过设置过滤规则的方法，按特定的主机及特定的协议端口如HTTP、FTP、Telnet、SNMP等进行过滤，便可以将网络管理员关心的敏感数据向管理模块提交。信息的过滤条件选择方法有如下几种[33, 42]：

1．服务过滤：根据端口筛选特定类型服务。

2．通用过滤：根据数据包中某一偏移处的16进制值选择特定数据包。

3．捕获前过滤：当不想缓冲区因无用的数据而溢出时这一方法很有用。

4．捕获后过滤：已经捕获了有问题的数据后还需进一步分析所有数据。

5．站过滤：根据MAC地址，筛选出某一个用户机的数据。

6．协议过滤：根据传输层和网络层中的特性过滤，如选择TCP数据而非UDP数据，或选择某一特定IP层协议数据。

3.5 协议分析效率

网络中流动的是二进制数据，网络监听程序必须能够对这样的二进制信息进行“解码”，从而分析出数据包的意义。在进行数据分析时应掌握必要的方法，以分析出关键信息和兼顾系统效率为出发点，只有这样工作才是有意义的，协议分析中应注意以下两个问题[34]。

1．在解析通过HTTP协议传输的数据包时，应注意不要将整个数据包的报文内容都提取并保存到数据库中，这样做不但会降低分析的效率同时会增加数据库的负载。解决的办法是在分析时，只提取URL信息并将之保存到数据库中，这样管理模块如果对某个HTTP信息感兴趣，便可以通过该URL信息进行访问[35]。

2．在解析通过FTP和TELNET协议传输的数据包时，只将登录名、口令和操作命令进行解析恢复并入库，并不对具体传输的报文数据或选项进行解析。

3.6 本章小结

本章首先给出了网络监听系统的一般结构及组成部分，然后对系统实现过程中的实时性与稳定性的考虑、系统性能的提高、过滤规则、协议分析效率等几个关键

问题进行了研究和阐述，指出了为了减少捕包过程中的丢包，提高监听系统的性能，必须解决好核心缓冲区和用户缓冲区的大小关系问题。研究表明，为了获得最好的性能，可以将应用程序的缓存设置成和驱动的核心缓存大小相同。

4 面向企业网络监听系统的设计与实现

在对网络监听的概念及网络数据包捕获技术深入研究后，本章具体讲述面向企业网络监听系统 Zjsniffer 的设计与实现。

4.1 监听系统运行环境

所设计的监听系统（Zjsniffer）主要面向企业局域网络，要求的环境为：系统平台是WindowsXP，捕获驱动程序采用的是WinPcap3.1，编译应用程序利用Visual C++6.0 。

4.2 监听系统的结构

监听系统分为两大部分:驱动程序部分和应用程序部分。驱动程序部分工作在核心态，负责网络数据的接收和发送，应用程序部分工作在用户态，与驱动程序进行正确的通讯外，缓冲区由应用程序动态分配。应用程序主要包括捕包模块、数据分析模块和管理模块。系统结构如图4.1所示。

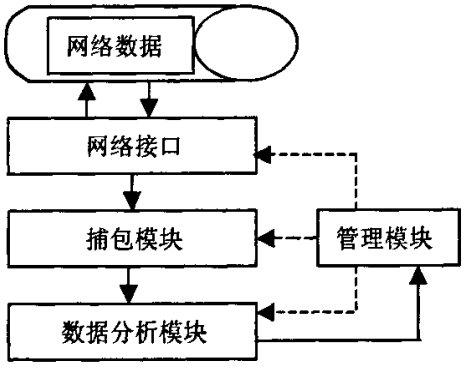

图 4.1 监听系统 Zjsniffer 结构图

1．捕包模块。捕包模块将网络接口设置为混杂模式(Promiscuousmode)，捕获网

络上的数据包，供协议分析模块使用。由于效率的需要，有时要设置过滤网上的数据包，如特定的IP，特定协议的数据包。网络监听模块的过滤功能的效率是关键，因为对于网络上的每一个数据包都会使用该模块过滤，判断是否符合过滤条件。低效率的过滤程序会导致数据包丢失、分析部分来不及处理等，为提高效率，数据包过滤应该在系统内核实现。利用WinPcap驱动程序，直接读取LAN网内的以太网数据帧，而不是在高层获得数据包。由于捕获程序工作在核心态，因此可以设置网卡的工作模式。

2. 数据分析模块。数据分析模块的主要功能是辩别数据包的协议类型并对数据进行适当的分析解释。可以把所有的协议构成一棵协议树，一个特定的协议是该树结构中的一个结点，见图4.2所示[36, 39]。一个数据包的分析就是一条从根到某个叶子的路径。树的结点数据结构中应包含以下信息：该协议的特征、协议名称、协议代号、下级协议代号、协议对应的数据分析函数链表。协议名称是该协议的唯一标志。协议代号是为了提高分析速度用的编号，如TCP的下级协议是IP协议。协议特征是用于判定一个数据包是否为该协议的特征数据，这是协议分析模块判断该数据包的协议类型的主要依据。

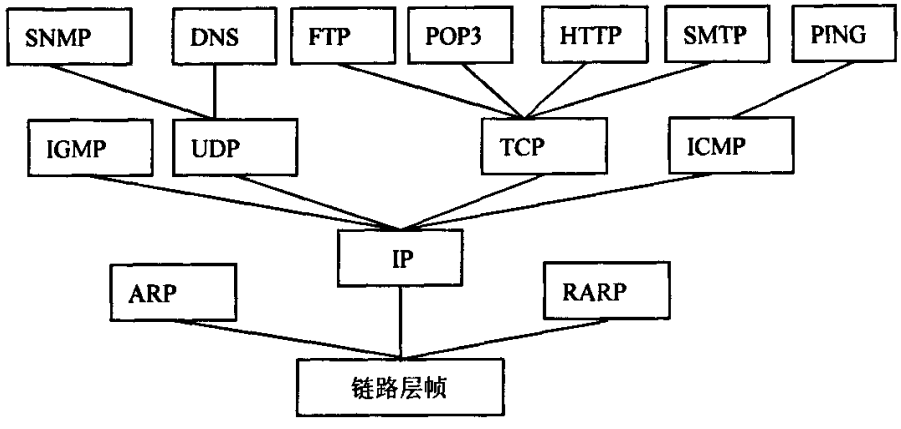

图 4.2　　协议树示意图

3. 管理模块。本模块的主要功能是用来统计和管理，用于与用户交互，将捕获到的数据进行流速等参数统计，并把当前用户通信的详细信息在用户界面上显示出

来。此外，通过本模块网络管理员还可以进行过滤规则的设置，该模块还同时协调捕包模块、数据分析模块的工作。

4.3 过滤规则的选择

在开发企业网络监听系统Zjsniffer时，根据需要综合运用了3.4节介绍的过滤方法。为了提高系统的效率和减少系统负担，对于所有的数据包都采用捕获前过滤，首先将不符合过滤规则的数据包拒之门外，然后综合运用协议过滤和服务过滤的方法。其中，过滤规则设置为:"tcp src port 80 or tcp dst Port 80 or tcp dst Port 21 or(tcp dst port 23 and less 55) or tcp dst Port 25"。这样只有使用HTTP、FTP、 Telnet或SMTP协议传输的数据包，捕获系统才对其进行采集，从而排除了许多并不关心的数据包。

由于同一个Telnet会话的数据包会很多，其中包括许多并不需要解析的报文 ，如选项协商报文等。通过分析，总结经验如下：缺省情况下TELNET登录时进入字符输入模式，对于用户名、口令和操作命令的输入,基本上是客户端一有击键就立即向服务器发送字符，TCP数据区就一个字节。在不考虑IP选项、TCP选项介入的复杂情况下，整个物理帧长度14十20十20+1=55，而其他的链接和协商报文，长度都不是55字节，这就很容易通过过滤规则过滤掉我们不关心的TELNET报文，可以加入附加过滤规则来减少捕获的数据包数量，以提高系统效率。如附加过滤规则：“tcp dst port 23 and less 55”。

对于FTP，其过滤过滤规则设置为tcp dst port 21，而不是tcp d st port21 and tcp dst port20，原因是只关心控制链接而不关心数据链接，所以为了提高系统效率，过滤掉了数据链接包，也即端口号为20 的数据，而对于HTTP，过则设置为捕获源端口号或目的端口号为80 的所有数据包 。

4.4 丢包问题的解决

在实现从驱动程序读取数据并过滤的过程中，需要解决的一个问题就是防止

WinPcap驱动程序丢包。通常的做法是：多开几个线程，每个线程都做同样的工作，发出读请求，然后等待读完成，然后实施过滤规则，每个线程都采取异步方式进行读。但线程开多了，例如当开两三个线程时，总是读到一些错误的数据。分析后发现：假设第一个线程发出读请求，然后被切换到第二个线程，而恰巧此时第一个线程的读请求已经完成了，而第二个线程线程调用得到第一个线程完成的结果，但实际上这时候第二个线程的读请求并未完成，所以第二个线程的buf里面没有任何数据，而第二个线程却误认为自己的读请求完成，就把数据传送给用户界面线程处理，丢包问题由此而产生其根本原因在于读是异步的。在考虑多方面的因素后，采用了使用两个线程的办法：一个线程{ReadRequestProc}不停的向驱动程序发送读请求，使得WinPcap的读队列始终不为空，当WinPcap收到数据时，总是有读队列在那里等待满足需求。这个线程不管读请求完成的情况，而是只管发出异步的读请求;另一个线程负责检查这些读请求完成的情况，每检测到一个读请求成功完成时，就对数据实施过滤规则，并决定是否通知用户界面线程处理。当然不能无限制的向WinPcap发送读请求，所以应采取一定的办法控制读请求队列的长度在一定的范围内，在程序中用(m-ReadQueueLength)来记录应用程序中当前已经发出但还没有检测到完成读请求队列的长度。这两个线程采用事件机制进行协调，保证读请求队列在一定的长度(不为空)。通过这样的处理，有效地解决了捕包过程的丢包问题。

此外，在程序设计中还要保证用户按下退出按钮时两个读请求线程的正确退出，所以应保证两个读线程中的另一个读线程先退出，然后再让当前的读线程退出并要考虑到内存的释放情况。

4.5 网络监听系统功能的具体实现

4.5.1 数据包的捕获

数据包是网络状态分析的主要数据来源，因此网络数据捕获是一个重要的模块。网络数据捕获模块是将在网络共享网段中，位于数据链路层的所有原始数据包捕获，

并可以保存在一个二进制文件中。要捕获数据包，就要求应用程序能够任意的启动和停止捕获驱动程序的服务，能够选择要用的网络设备(网卡)进行监听。

Zjsniffer的捕包系统采用WinPcap，捕包流程见图4.3所示。

在初始化完毕后，系统利用WinPcap的pcapesloop()捕获数据链路层的数据包，然后根据数据链路层类型选择数据包，由packet.dll上传至用户区，进行协议分析，实施信息匹配。

系统设计中用到的12个WinPcap，主要函数具体功能为:[38, 39, 40, 41]

(1) 函 数 pcap_t*pcap_open_live(char*device,intsnaplen,int promisc,intto_ms, char*ebuf)

该函数用来获取一个包捕获器描述符，这是最重要的一个函数。首先，它打开网络适配器，把网卡设置成"混杂"模式，使它能够接收来自网络的所有数据包;其次，它为应用程序设置一个缺省大小为256kB的缓冲器;最后，它为包捕获驱动器分配一个缺省为1MB的内核缓冲器，可以根据需要，改变该缓冲器的大小。

(2) 函数char*pcap_lookup dev()

该函数返回一个适于Pcap_open_live()和pcap_look-upnet函数使用的指向网络设备的指针。

(3) 函数int pcap_lookup_net() 该函数用于判断与网络设备相关的IP地址和掩码。

(4) 函数int pcap_compile()

用于将过滤规则字符串编译成一个内核过滤器。

(5) 函数int pcap_stefilter(pcap_tp , s ruct bpf_program*fp))

该函数用来设置内核过滤器，这两个参数的意义如前述,过滤器程序则在一个名为bpf program的结构中定义。

(6) 函数pcap_loop(pcap_t*p,int cnt,pcap_handler call back,u_char* user)

该函数捕获并处理数据包。cnt指定函数返回前所处理数据包的最大值，值为负时表示处理缓冲区里的所有数据包，为0时表示处理所有数据包，直到产生读到EOF或者超时读取错误时停止。callback指定带3个参数的回调函数。use为用户传递给回调函数的指针，通常为NULL。

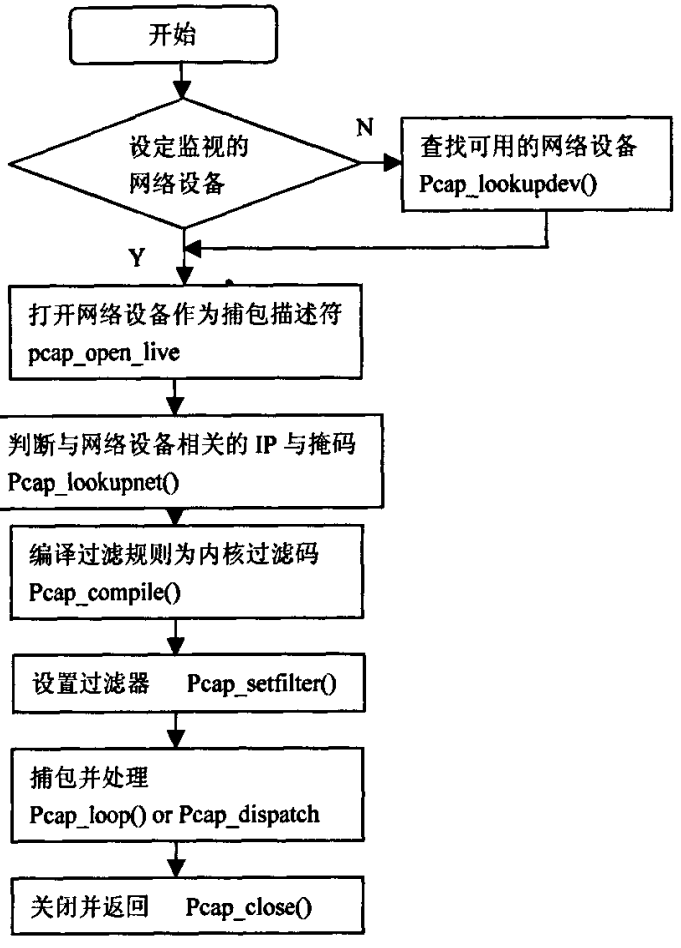

图 4.3 捕包流程图

(7) 函数void pcap_close()

关闭关联文件并回收资源。

(8) 函数void callback(u_char*args,const struct pkthdr header,const u_char* packet)

(9) 函数Pcap_setbuff

该函数用来设置包驱动器缓冲器的大小，一个适当大小的缓冲器不仅可以减少丢包率，还可以提高包捕获的速度。

(10) 函数Pcap_setmode

用于把网络适配器设置为统计方式。

(11) 函数Pcap_stats

用于获得包捕获过程的统计数据。

(12) 函数Pcap_read

从包捕获驱动器中读取一组数据包并针对每一个包运行包过滤程序，然后把过滤后的数据送应用程序缓冲器。

通过以上WinPcap函数，进行数据数据包捕获模块程序设计，其中用户对数据包的检查或处理程序可以通过callback调用。

4.5.2 协议分析与数据解释

协议分析与数据解释的目的是根据定义好的数据结构，从原始的网络数据包中解析出协议信息。按照协议栈自下向上的顺序调用，从数据链路层到传输层、网络层和应用层，以使相应的数据解析程序来检测数据包。

为了将捕获得到的数据解析出来，必须对TCP/IP协议进行深入的研究。数据进入TCP/IP协议栈的封装过程,如图4.4所示。具体的数据解析过程是数据帧封装过程的逆过程，这个过程需要对多种需要的帧格式加以分析。

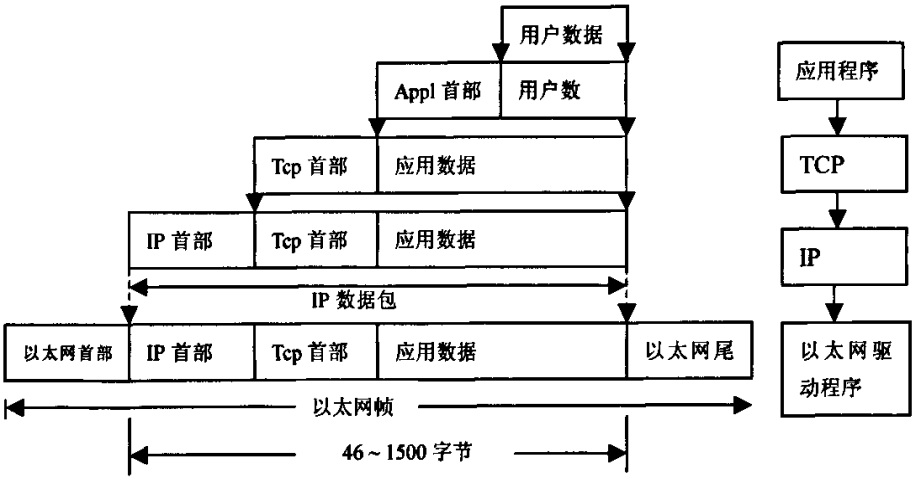

图 4.4 数据进入协议栈的封装过程

(1) 数据帧类型的判断

IEEE802帧格式如下(RFC1 042)，如图4.5所示。

802.3MAC				802.2LLC		802.2SNAP			
目的地址	源地址	长度	DSAP AA	SSAP AA	ONTL 03	DRG CODE 00	TYPE	DATA	CRC
6	6	2	1	1	1	3	2	38-1492	4

图 4. 5 IEEE802 帧格式

ETHERNETII帧格式(RFC849)，如图4.6所示。

目的地址	源地址址	长度	数据	CRC
6	6	2	46~1500	4

图 4.6 ETHERNETII 帧格式

ETHERNETII帧格式封装是事实标准，在捕获的数据包中基本上都是ETHERNETII格式的数据帧。

判断帧类型的简易算法

```
If 帧类型/帧长度 <= 1500 (十进制)
    If 帧类型/帧长度字段后继第一个word是FFFF(十六制),此Packet是raw
       ETHERNETII802.3
     else
      if 帧类型/帧长度字段后继第一个word是AAAA(十六进制) and is 03(十
      六进制)
          此 Packet是ETHERNET SNAP
        else
          此Packet是 ETHERNET 802.2
       end if
    end if
else
        此 Packet是ETHERNETII
end if
```

(2) IP、ARP及RARP协议类型的判别

用EthernetII帧格式的类型域可以区分IP、ARP及RARP协议。类型字段的意义(RFC1700)，如表4.1所示。

表4.1 以太网帧类型字段意义

类型（十六进制）	协议
0800	IP
0800	ARP
8035	RARP

WinPcap 在WinXP操作系统下捕获的数据包没有4个字节的CRC校验部分，所以整个帧的长度是60-1514字节。

ARP(地址解析协议)的帧格式，如图4.7所示。

以太网目的地址	以太网源地址	帧格式	硬件类型	协议类型	硬件长度	协议长度	操作码	发送者以太网地址	发送者IP地址	目标以太网IP地址	目标IP地址

图 4.7 ARP 帧格式

RARP(逆地址解析协议)的帧格式与ARP的帧格式是相同的，只是ARP帧类型域中的数值是0806，RARP帧类型域中的数值是8035，这样数据结构树的第一层数据便解析出来。

对第二层的数据(即IP数据)进行进一步解析。IP(网际协议)格式如图4.8所示。

4位版本号	4位报头长度	8位服务类型	总长度（字节） 16位	
16位标识			3位标志	13位分段位移
8位生存期	8位协议号		16位报头校验	
32位源IP地址				
32位目的IP地址				
数据...				

图 4.8 IP(网际协议)格式

在IP数据包的基础上进行数据解析是根据协议号域中的值来判断的，进行如下区分，如表4.2所示。

表 4.2 按照 IP 协议号区分帧格式

协议号（十六进制）	协议类型
01	ICMP
02	IGMP
06	TCP
11	UDP

（3）UDP、TCP协议类型的判别

根据数据包头长度将IP包头剥离后即为剩余的数据，对第三层即传输层的数据(即UDP或TCP数据)进行进一步解析，其中UDP协议对数据的封装格式为如图4.11所示[36]。

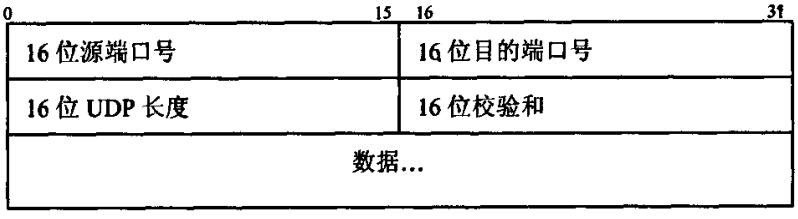

图 4. 11 UDP 格式

由此看出UDP报文头部固定长度为8字节。

TCP协议的格式如图4.12所示。

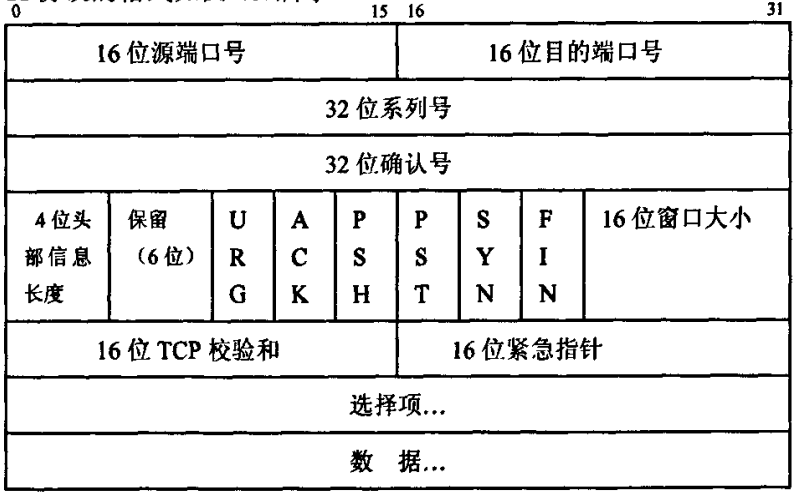

图 4. 12 TCP 格式

由图看出 TCP 报文段头部固定长度为 20 字节，在有选择项数据时，头部长度不固定，可从 4 位头部信息长度域读出（以 4 字节为计算单位）。

TCP 与 UDP 通过端口号为解析数据提供依据。由于端口号可达一万多种，在对应用层协议进行判别时，一般仅对常用的 FTP 协议、HTTP 协议以及 Telnet 协议的数据进行解析。TCP 与 UDP 的端口分配基本上是相同的，主要端口如表 4.3 所示[37]。

表4.3 常用端口含义

十进制数	关键字	描述
20	ftp(default data)	文件传输协议（默认数口）
21	ftp(control)	文件传输协议（控制）
23	telnet	TELNET
80	www.http	万维网（http）
25	Smtp	简单邮件协议

（5）协议头结构的设计

根据报文的结构，设计一个相应的协议头结构

* struct ETHERNET_HEADER

{byte deses_mac[6]; //接收端的MAC地址

byte src_mac[6]; //发送端的MAC地址

byte type[2]; //类型字段

};

* struct IP_HEADER

{byte ver_len; //版本4位，头长度4位，报头长度以32位为一个单位

byte type; //类型8位

byte length[2]; //总长度，16位，指出报文的以字节为单位的总长度，报文长度不能超过65536个字节，否则认为报文遭到破坏

byte id[2]; //报文标示，用于多于一个报文16位

byte flag_of set[2];//标志,3位数据块偏移43位

byte time; //生存时间，8位

byte protocol; //协议，8位

```
byte crc-val[2];  //头校验和，16位
byte src_addr[4]; //源地址，32位
byte des_addr[4]; //目标地址，32位
byte options[4];  //选项和填充，32位
};
* struct TCP_HEADER
{byte src_port[2];      //发送端端口号，16位
byte des_port[2];       //接收端端口号，16位
byte sequence_no[4];  //32位，标示消息端的数据位于全体数据块的某一字节的数字
byte acke_no [41];  // 32位，确认号，标示接收端对于发送端接收到数据块数值
byte offset_reset_con[2]; //数据偏移4位，预留6位，控制位6 byte
window[2];                  //窗口16位
byte check_sum[2];          //校验码，16位
byte urgen_pointer[2];     //16 位，紧急数据指针
byte options[4];            //选样和填充，32位
};
struct UDP HEADER
{byte src_port[2];          //发送端端口
byte des_port[2];           //接收端端口
byte length[2];             //用户数据包长度
byte checksum[2];           //校验码
};
* struct ICMP HEADER
{byte type;                 //类型字节(1字节)
byte code ;                 // 代码字节 (1字节)
byte checke_sum[2];         //校验码(2字节)
```

```
};
* struct ARP_PROTOCOL
{byte hw_type [2];          //硬件类型。以态网表示为1
byte protocol[ 2 ];         //网 络层协议的类型，例如IP协议为800
byte hw_len ;               //查询物理地址的字节长度,以态网时为 6
byte protocol_ len ;        // 查询上层协议地址的字节长度,IPv4为4
byte opcode[2 ];            //表示操作内容的数值,1为ARP请求，2为ARP响应,3为RARP请求4为RARP响应
byte src_ mac[ 6];          //发送端MAC地址
byte src_addr[ 4];          //发送端的IP地址
byte des_mac[ 6];           // 查询对象的MAC地址
byte des_addr[ 4];          //查询对象的IP地址
};
```

(6) 协议规范指出以太网数据包中第13字节处包含了两个字节的第三层协议标识，利用这个规范，分析协议分析模块:

① 跳过前面的第12个字节，读取13字节处的2字节协议标识0800 。根据协议规范可以判断该网络数据包是IP包。

② IP 协议规定IP包的第24个字节处有一个1字节的第四层协议标识因此系统跳过15到24字节直接读取第四层协议标识：06，该数据包是TCP协议。

③ TCP 协议在第35字节处有一个2字节的应用层协议标识(端口号),于是系统跳过第25到34字节,直接读取到第35字节的端口号：80。该数据包是一个HTTP协议的数据包。

从网卡那里捕获原始数据包后返回链路层的首指针，然后数据链路层协议分析函数就被调用，每个协议分析程序工作方式都很相似，填写Packet结构的相应值，并提交给网络层协议分析函数，网络层协议分析函数处理后再提交给相应的传输层协议分析函数，见图4.13。

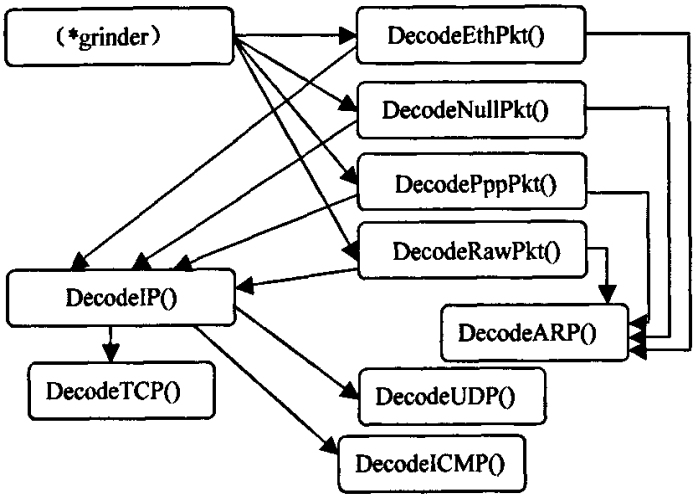

图 4.13 协议分析

4.6 本章小结

本章介绍了网络监听系统 Zjsniffer 运行的环境和基本结构，阐述了在系统设计过程中解决丢包问题及过滤规则的选择等关键问题的实现方法，给出了捕包过程的流程图，对设计中用到 WinPcap 的 12 个函数的具体功能做了详细的说明，对捕包模块的实现作了详细的阐述。同时，在对 TCP/IP 协议进行深入研究的基础上，从数据封装过程、数据帧类型的判别入手，介绍了数据分析与协议分析模块的实现过程。

5 系统性能测试

5.1 系统性能测试实验

1. 测试目的：对所设计的监听系统 Zjsniffer 的各项性能进行测试，然后根据测试的结果给出简要说明。

2. 主要测试项目

(1) 捕获数据包的数目及协议种类

(2) 数据包内容及分析

(3) 数据流速统计

(4) 协议过滤功能

3. 测试环境

(1)测试地点：广西柳州天虹广告设计公司设计室

(2)硬件环境：18 台通过 HUB 连接的 PC 机，P4，主频为 2Ghz

(3)PC 机的操作系统为 WinXP，安装 WinPcap3.1，测试程序为所开发的网络监听系统 Zjsniffer。

5.2 测试结果

1. Zjsniffer 捕获数据包的基本界面，见图 5.1

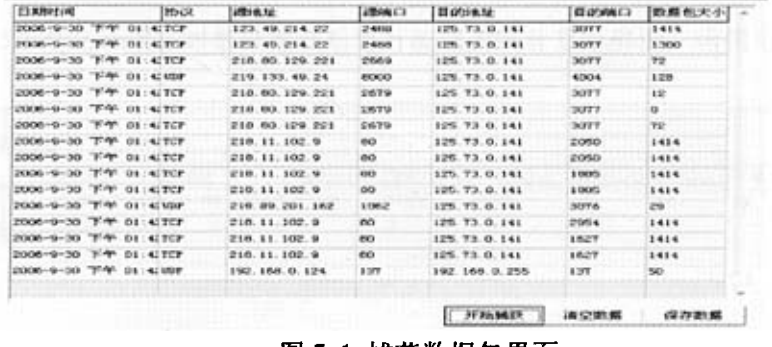

图 5.1 捕获数据包界面

显然，Zjsniffer 能实时地捕获网络中流经的数据包，能对数据捕获的时间、源地址、源端口、目的地址、目的端口、数据包大小进行纪录，同时能对数据包的大小及协议类型进行分析和显示。

2．数据包内容及分析

Zjsniffer 分析数据包的基本界面，见图 5.2

图 5.2 数据包内容及分析

图5.2显示协议分析模块能显示出数据的实际内容（十六进制），显示出数据包源IP地址、目的IP地址等相关字节内容，该模块还可根据帧类型判别算法和帧构成规则，对捕获到的数据进行逐个字节的分析并显示出分析结果。网络管理员可以通过这部分功能实现对PC用户上网的监控。

3．数据流速统计

图 5.3 是 Zjsniffer 对数据捕获后进行流速的统计显示界面，由图可以

图 5.3 数据流速统计界面

观察到 Zjsniffer 监听系统能对所捕获的数据的流入和流出速度等内容进行统计，通过对数据流速的监控，企业网络管理员可以了解网络数据流动情况，及时发现网络流量异常，例如通过检测到的网络流量异常来判断网络是否遭受攻击等等。

4．Zjsniffer 协议过滤能力

由图5.4可见，Zjsniffer可以有选择地对流入的数据包进行协议过滤，该功能可以使企业网络管理员灵活地对网络流入流出的数据类型进行监控。

图 5.4　协议过滤

5.3 本章小结

本章对所设计的监听系统（Zjsniffer）在企业网络中心进行了测试运行，并从运行的测试界面分别对系统的数据包捕获与分析能力、协议分析与过滤能力、流量统计等几个方面进行了分析，分析结果表明（Zjsniffer）具备一定的网络监听功能，是企业网络管理员的得力助手。

6 总结与展望

6.1 总结

本文简单介绍了网络数据包及网络数据包捕获的概念和基本方法，对 WinPcap 的体系结构、包捕获驱动机制等内容进行了深入的理论研究，并对利用 WinPcap 进行网络数据包的捕获和过滤进行了试验研究，并以其在网络安全监控系统的实际应用，阐述了这一技术的具体应用细节，开发了具有一定功能的网络监听系统。

Windows 网络数据包捕获技术作为Windows平台网络信息安全系统的底层核心技术，是设计和实现各种Windows平台网络信息安全系统的基础。长期以来，由于Windows源代码的不公开和国外各大IT公司为了商业利益而对这类核心技术的封锁，使得目前国内能够真正掌握这一技术并形成实际可应用的产品并不多见。从目前的情况来看，包嗅探实现网络数据监听也有专用硬件和软件的产品，尖端的网络分析仪产品甚至可以找出一般网络接口检测不到的错误。在实际的应用中，选择合适的实现方式来构建系统的底层网络数据包拦截模块尤为重要，必须结合系统需求和所需代价综合考虑。

面向企业局域网络监听系统的设计思想实现了对局域网环境下机房监控，通过本监听系统管理员可以监视网络的状态、数据流动情况以及网络上传输的信息，并可以利用这些信息来排除网络故障。由于监听程序不必与网络上的关键设备交互，避免了对网络性能的影响，而且可能获得比较敏感数据，这样使得管理人员可以监控人为的消息泄漏，对网络安全也具有重要意义。

6.2 展望

通过两年的理论学习和专业研究实践，本人对包捕获基本原理及网络监听技术有了较深入的了解和认识，对于本研究项目，在现有工作的基础上，本人认为下一步应该在以下几个方面开展研究：

1. 系统的功能和统计模块还显得比较单薄，对包协议的分析显得不够详细和可靠，尚不能满足用户多方位多条件的统计要求。在后续工作中，可以考虑为管理统计模块多设置一些统计条件供用户选择，并且能以多种方式向用户显示结果，比如说数据包流量，系统可以考虑设计给出柱状图的界面。

2. 可以考虑增加对邮件内容的过滤功能，同时还要考虑跨网段的用户监控问题。

3. 后续工作还要开发系统的攻击检测模块，对于这一模块需要实现一个攻击规则库，其中规则的分类是很重要的，对于由多个数据帧系列组成一次有效攻击的情况要多花精力去研究。

4. 在包捕获原理上，除了继续研究基于WinPcap的包捕获方法，还计划学习零拷贝技术的高速网络数据包捕获技术。

致 谢

本论文是在导师徐兰芳副教授的悉心指导下完成的。徐兰芳老师与作者就论文选题和论文内容进行了详细、深入的讨论，给予作者启发性的指导并提出了有益的建议和帮助。作者在学习期间，深刻地感受到徐兰芳老师严谨治学的学术风范，这将对我以后从事科研工作会有重大的影响。在此，我衷心感谢导师徐兰芳老师的指导与帮助!

同时我要感谢在百忙中抽空对我的论文进行阅读和评阅的老师，感谢他们对我的研究工作和论文撰写提出了宝贵的意见和建议!

感谢广西工学院能够给予已经走上工作岗位上的我这样一个学习的机会!

感谢所有帮助过我的老师、同学们!

参考文献

[1] 谢希仁. 计算机网络. 第三版. 北京:清华大学出版社，2003 年 8 月. 10~62

[2] 洪帆. 信息安全概论. 北京:高等教育出版社，2005 年 8 月. 6~22

[3] Charlie Kaufman, Radia Perlman, Mike Speciner, Network Security: Private Communication in a Public World, Second Edition, 2002年5月. 20~26

[4] 钱丽萍, 高光来. 包捕获技术:原理、防范和检测. 计算机系统应用, 2002 年, 第 2 期:31~33

[5] 姚冒群. 网络攻击的分析及防范策略. 计算机应用研究, 1999 年，第 4 期:34~35

[6] 赵粮, 裘晓峰. 现代入侵检测技术. 学术与技术, 2001 年，第 1 期:31~33

[7] 邓瑛, 常国岑, 王晓辉. 网络安全监控与审计系统的设计与实现. 计算机工程, 2002年, 第28卷第12期:195~198

[8] 胡华平, 陈海涛, 黄辰林, 唐勇. 入侵检测系统研究现状及发展趋势. 计算机工程与科学, 2000 年, 第 25 卷第 2 期:20~25

[9] Varenni, G. Baldi, M. Degioanni, L. Risso, F. Optimizing packet capture on symmetric multiprocessing machines. Computer Architecture and High Performance Computing, 2003. Proceedings. 15th Symposiumon 10-12 Nov. 2003 Pages:108~115

[10] Risso, F. Degioanni, L. An architecture for high performance network analysis. Computers and Communications, 2001. Proceedings. Sixth IEEE Symposiumon, 3~5 July 2001. Pages:686~693

[11] 孙华, 曹袖. 分布式网络入侵监视系统的设计. 计算机工程，1999 年, 第 25 卷第 3 期:60~62

[12] 韦卫, 王德杰. INTERNET 网络层安全协议理论研究与实现. 计算机学报，1999 年, 第 4 期:31~35

[13]刘炎,冯穗力,叶梧,徐宇强.WDM/NDIS 网络驱动程序实现方法的研究.计算机应用研究,2001 年, 第 8 期:118~120

[14]钱丽萍,高光来.包捕获技术、原理、防范和检测.计算机系统应用,2002 年,第 2 期:31~33

[15]赵心宇,朱齐丹,朱达书.应用 winPcap 捕获网络数据包.应用科技,2004 年,第 11 期:29~31

[16]龚剑,冯春.基于网络报文的网络瓶颈带宽测试技术的研究.计算机工程与科学,2001 年,第 1 期:1~6

[17]张承,蒋东兴,刘启新等.浅析网络监控系统对网络性能的影响.小型微型计算机系统,2002 年, 第 23 卷第 9 期:1059~1062

[18]易发胜,彭孜,李成林.Windows 的网络体系结构分析及网卡驱动程序设计.计算机应用,1999 年,第 19 卷第 10 期:61~63

[19]高翔,苏广文,胡正国.入侵网络系统中的网络监测.微电子学与计算机,2002 年,第 2 期:37~39

[20]曹卫兵,王安,敬忠良等.网络安全与防火墙技术.通信技术,2001 年,第 6 期:27~29

[21]张明武,肖宏年,邹晓.基于 NDIS 的网络检测与分析.湖北工学院学报,2001 年,第 16 卷第 1 期:14~16

[22]吴勇军.用 WinSock 编程捕获 IP 包.计算机工程与设计,2004 年,第 25 期第 5 卷:691~693

[23]郑啸,魏仰苏.一种新的面向协议测试的包捕获结构.华中科技大学学报(自然科学版),2004 年,第 32 卷第 7 期:16~18

[24]Synthesis and analysis of fractal LAN traffic at high speeds. Rincon, D. Martinez, S. Cano, C. Sallent, S. Local and Metropolitan Area Networks, 2004. LANMAN 2004. The 13th IEEE Workshop on , 25~28 April 2004 :Pages:259 ~264

[25]E. Jonsson and T. Olovsson. A quantitative model of the security intrusion

process based on attacker behavior. IEEE Transactions on Softw are Engineer ing. 1997, Vol23, No4: 687~694

[26]李雪莹,刘宝旭,许榕生.基于 winPcap 的网络监控系统性能优化.计算机工程,2004 年,第 30 卷第 1 期:8~9

[27]G. Post, A. Kagan. An evaluation of Rose. Information and Software Technology. Volume:42, Issue:6, April15, 2000: 33~38

[28]Hala ElAarag, Bassiouni, Mostafa. Performance evaluation of TCP connections in ideal and non-ideal network environments. Computer Communications. Vol. 24, Issue:18, February 2001:1769~1779

[29]李洪海,龚世生.网络监控的基本原理和标准介绍.现代计算机,1999 年,第 77 期:56~59

[30]张戈,张敏华,何晓.网络监听技术在局域网中的实现.微机与应用,2002 年,第 2 期:3~8

[31]陈科,李之棠.网络入侵检测系统和防火墙的集成模型.计算机工程与科学,2002 年,第 23 卷, 第 2 期:26~28

[32]梁理,黄樟钦,侯义斌.网络信息侦听系统的研究与实现.计算机工程应用,2002 年,第 7 期:184~187

[33]吕坤,钟宏.在 Win32 环境下开发高性能网络监测程序 . 中国计算机协会计算机体系结构学术年会,2002:196~200

[34]庄春兴,彭奇志.基于 Winpcap 的网络嗅探程序设计.计算机与现代化年,2002 年,第 5 期:11~13

[35]许士博,颜学雄,王清贤.Windows NT 下开发网络监控程序.信息工程大学学报,2000 年,第 3 期:31~37

[36]刘志雄,贺贵明,宋志伟.Ethernet 网络监控的原理及其在 Windows 环境下的编程实现.武汉大学学报(工学版),2002 年,第 35 卷第 1 期:102~105

[37]GR. Mallan, F. Jahanian. An Extensible Probe Architecture for Network Protocol Performance Measurement. Proc. ACM SIGCOMM. 1999

[38]徐兰芳,潘芸.数据仓库安全需求模型研究.华中科技大学学报,2005年,第33卷第7期:8~12

[39]周世兵,刘渊.多层次的内部网安全策略研究及应用.计算机应用研究,2002年,第9期:129~133

[40]唐正军,刘代志.网络嗅探器sniffer软件源代码浅析.计算机工程,2002年,第2期:38~41

[41]http://www.sniffer.com

[42]http://www.securitfocus.com

附录A 利用WinPcap进行网络数据包捕获和过滤实验源代码

实验的主要源代码及说明

根据用 Windows 分组捕获库 WinPcap 提供功能首先要初始化两个结构体，一个是适配器的结构体 LpAdapter，一个是存放接收到的数据包的结构体 RecvPacket。

使用 Packet.dll 动态链接库编写源代码：

```
#define MAX__LINK__NAME__LENGTH 64
```

适配器结构：

```
typedef struct__ADAPTER
{
HANDLE hFile;
TCHAR Symboliclink[MAX__LINK__NAME__LENGTH];
Int NumWrites;
} ADAPTER ， *LPADAPTER;
```

说明：hFile 是一个指向该网络适配器 Handle 的指针，通过它可以对网络适配器进行操作。symboliclink 包含当前打开的网络适配器的名字。

数据包结构：

```
typedef struct _PACKET
{
HANDLE          hEvent;
OVERLAPPED      OverLapped;
PVOID           Buffer;
UINT            Length;
PVOID           Next;
UINT            ulBytesReceived;
BOOLEAN         bloComplete;     //控制接受包的开始和结束
```

数据包捕获实现的步骤的主要源代码：

1. 要获得适配器列表。

```
#define     Max__Num__Adapter 10      //获得适配器列表
char        Adapterlist [Max__Num__Adapter[512]]
int         i=0
```

```
char        AdapterNames[512],  *tempa, *templa;
ULONG AdapterLength=1024; } PACKET,  *LPPACKET:
```

2. 获得系统中网络适配器的名字。

```
PacketGetAdapterNames(AdapterNamea, &AdapterLength);
tempa=AdapterNamea;
templa=Adapternamea;
while ((*tempa!='\0') | | (*tempa-1!='\0))
{
if (*tempa= ='\0')
{
memcpy(AdapterList[i], temla, tempa-templa);     //内存数据拷贝
templa=tempa+1;
i++
}
tempa++
}
```

3. 从适配器列表中选择一个默认的 0 号适配器。

```
LPADAPTER lpAdapter
lpAdapter = PacketOpenAdapter (AdapterList[0]);
if (!lpAdapter | | (lpAdapter->hFile==INVALID__HANDLE__VALUE))
{
dwErrorCode=GetLastError();
return FALSE;
}
```

4. 将所选择的适配器 lpAdapter 设置为混杂模式。

```
PacketSetHwFilter(lpAdapter, NDIS_PACKET_TYPE_PROMISCUOUS)
```

5. 设置 BPF 内核中包过滤的过滤器的代参政。利用这个函数右以完成对于原始数据包的初始的过滤处理，如根据其中端口号、IP 地址等。

```
PacketSetBpf（LpAdapter AdapterObject, structbpf_program*fp）
```

6. 设置缓冲池为 5 1 2 K字节。

```
PacketSetBuff(lpAdapter, 512000);
```

7. 分配一个数据包对象，并链接已分配的缓冲。

PacketInitPacket(lpPacket，(char*)bufferReceive，512000);

8. 捕获多个数据包。从网卡 lpAdapter 接收数据包，并将数据包放入 lpPacket 所指向的数据包结构体中，若接收成功返回 TRUE，否则返回 FALSE。

PacketReceivePacket(lpAdapter，lpPacket，TRUE);

9. 通过触发回调函数，把捕获符合过滤器规则的数据包转发给网络协议分析模块进行分析处理。

10. 结束接收数据包，释放数据包对象。

```
if(lpPacket!=NULL
{
PacketFreePacket(lpPacket);
lpPacket=NULL;
}
```

11. 关闭网卡设备，将网卡恢复到正常接收状态。

```
if(lpAdapter!=NULL
{
PacketCloseAdapter(lpAdapter);
lpAdapter=NULL;
}
```

附录 B　WinPcap 常用的数据结构

1. pcap_t

本结构是WinPcap库的核心数据结构，基本上在程序的开始都要调用pcap_open_live()，该函数返回指向这个结构的指针，几乎调用库里的每一个函数都要这个结构作为参数。

(1)编程时需要涉及到的成员

编程时需要涉及到的成员有：打开设备的描述符，在WIN32平台下是ADAPTER *adapter和LPPACKETPacket;在LINUX/UNIX平台下是int fd ，指向所捕获到数据的缓冲区指针:u_char*bufer。

(2) 结构的具体定义

结构typedef struct pcap pcap_t, pcap的具体定义如下:

```
struct pcap {
#ifdefWIN32
ADAPTER *adapter;
LPPACKETPacket;
int fd ;
int snapshot;
int linktype;
int tzoff;     /*timezone offset*/
int offset;    /*offset for proper alignment*/
struct pcap_sfsf;
Struct pcap_md md; /*Read Buffer*/
int bufsize;
u char *pkt;
u_char* bp;
```

```
int cc; /**Place holder for pcap_next()*/
u-char*pkt; /*Place holder for filter code if bpf not in keneL*/
struct bpf_program fcode;
char errbuf[PCA_ERRBUF_SIZE]; };
```

2. 结构struct pcap_stat

该结构包含两个可用成员(目前所有平台均不支持另一个数据成员u_int ps_ifdrop)。具体结构如下：

```
struct pcap stat {
u_int ps_recv; / *numberof packets received*/
u_int ps_ drop ; /*numberof packets droped*/
u_int ps_ ifdrop ; /* drops by interface XXX not yet supported* / } ;
```

3. struct bpf_hdr和struct pcap_pkthdr

内核过滤器每输出一个包，将在输出的数前添加加字节的数据，这就是struct bpf_hdr。具体结构如下：

```
struct bpf_hdr {
struct timeval bh_tstamp;/ *time stamp*/
u_int   bh_caplen; /*length of captured portion*/
u_int   bh_datalen; /*original length of packet*/
u_short      /*length of bpf header( this struct plus alignment pad di ng )*/    };
```

4. struct pcap_pkthdr结构

在每一个被捕获的数据包存储到文件中时，结构struct pcap_pkthdr被加在包头。结构具体如下：

```
Struct pcap_pkthdr{
Struct timeval ts ; /* 接收此包的时间戳*/
Bpf_u_int32_caplen ;/*捕获数据长度*/
Bpf_u_int32len;      / *原始包长度*/ };
```

5. struct bpf_program

此结构如果使用tcpdump表达式，则不用关心其内部结构，但如果要手工编写BPF程序，就需要知道其成员。具体结构如下：

```
struct bpf_program{
u_int bf_len;
struct bpf_insn* bf_insns; }
```