

文章编号:1001-9081(2007)08-1858-04

一种 Linux 平台上基于包过滤的网络流量采集系统

聂朝恩,高荣芳

(西安石油大学 计算机学院, 西安 710065)

(chnnie@xysu.edu.cn)

摘 要:设计并实现了一种 Linux 平台上基于包过滤的网络流量采集系统 PFC。PFC 系统主要通过在内核空间实现数据包的过滤、合并,以及实现了用户空间和内核空间的内存共享,从而突破了传统上基于包过滤网络流量采集系统的性能瓶颈。

关键词:网络性能;流量采集;包过滤;Linux;数据流

中图分类号: TP393.06 **文献标志码:** A

Network traffic collection system based on packet filtering on Linux

NIE Chao-en, GAO Rong-fang

(School of Computer Science, Xi'an Shiyu University, Xi'an Shaanxi 710065, China)

Abstract: The design and implementation of a Linux-based packet filter network traffic collection system named Packet Flow Collector (PFC) was proposed. Data packet filtering and merging were realized in kernel space, and memory sharing between user space and kernel space was implemented, so that PFC system made a breakthrough in the performance bottlenecks of traditional packet filtering network traffic collection system.

Key words: network performance; traffic collection; packet filtering; Linux; packet flow

0 引言

网络流量信息采集是许多计算机网络设备或网络应用的重要功能,尤其在网络安全、网络管理和网络测量等专业领域的产品中,更是处于基础性的、不可缺少的地位。例如,防火墙、认证计费系统、入侵检测系统、信息审计系统、网络流量分析系统以及各种网管软件,其工作原理都是首先要从网络接口处采集网络流量信息,然后再进一步对采集到的数据进行分析处理。另一方面,近年来,Linux 操作系统因其开源、简单易用等特点得到广泛应用,并且成为许多网络产品的开发平台。因此,研究与开发在 Linux 平台上的网络流量采集系统具有重要意义。

目前 Linux 平台上的网络流量采集系统可分为两大类:1)基于 SNMP MIB 的采集系统,即在支持 SNMP 协议^[7]的网络环境中部署多个管理者(Manager)和代理(Agent),管理者通过代理收集经过各个网络设备的流量信息,存入 MIB,然后传递给相关网络应用访问,但是这种采集系统适用带宽较低并且会增加网络设备负载,影响其正常性能;2)基于在线数据包(packet)过滤的网络流量采集系统,即在 TCP/IP 网络协议层截获数据包,再传递给上层的网络应用,这一类采集系统适用面广,实时性高,且不会增加额外网络负载,因而在高速 IP 网络流量采集中被广泛应用。本文重点研究后者。

传统上,基于在线数据包过滤的网络流量采集系统的实现框架为:1)在以太网的出口处,例如在交换机和路由器之间,架设一台装有 Linux 操作系统的主机或其他网络设备作为网关;2)在网关的 Linux 内核中增加网络设备驱动模块,通过修改 TCP/IP 协议栈来截获流经网络接口的数据包,暂存于内核缓冲区中;3)在网关的用户空间开发一个 daemon 程序频

繁地执行 IO 中断调用,从内核缓冲区中读出数据包给用户空间应用程序,例如直接执行 IOCTL 系统调用命令,或者执行定制的 Socket 接口调用,或者间接地调用第三方开发包如 Libnet/Libpcap^[8]提供的 API;4)网关用户空间应用程序对接收到的数据包进行过滤、合并,得到一个数据流(Packet flow),然后发送给集中处理网络流量信息的数据服务器。在以上实现框架中,数据包从内核缓冲区拷贝到用户空间是通过频繁执行 IO 中断调用来完成的,然而,IO 中断调用一般要进行多次缓冲区数据拷贝,开销比较大,不仅影响网络转发的性能,而且当网络流量增大到一定阈值时,IO 调用获取数据包的速度就会低于操作系统内核中数据包流经网络接口的速度,导致一部分数据包从内核缓冲区中“溢出”而无法被采集到用户空间,从而降低网络流量信息采集的准确性。

从以上分析可以看出,内核空间到用户空间的数据拷贝是基于在线数据包过滤网络流量采集系统的性能瓶颈。为此,本文设计并实现了一种 Linux 平台上基于包过滤的网络流量采集系统 PFC(Packet Flow Collector),该系统在传统实现框架上有以下改进:1)把对数据包的过滤、合并操作从网关的用户空间迁移到操作系统内核空间,使得每次从内核空间读取的不再是一个数据包,而是过滤、合并处理后的数据流,从而大大减少拷贝的数据量;2)在机制上,内核空间到用户空间的数据拷贝不再通过 IO 中断调用,而是通过共享内存的方式来实现,即通过虚拟文件系统实现用户空间和内核空间的内存映射,使得用户空间的 daemon 程序可以直接访问内核缓冲区,不再需要中间的缓冲区拷贝。经过实验证明,PFC 系统对网关设备的性能影响很小,而且显著提高了网络流量信息采集的准确性。

收稿日期:2007-03-01。

作者简介:聂朝恩(1975-),男,河北定州人,工程师,硕士,主要研究方向:管理信息系统、计算机网络;高荣芳(1963-),女,陕西韩城人,副教授,硕士,主要研究方向:管理信息系统、计算机网络。

1 PFC 网络流量采集系统的总体结构

如图1所示,PFC网络流量采集系统由内核空间部分和用户空间部分构成。在内核空间,系统初始化时分配一块足够大的数据缓冲区,然后围绕内核缓冲区,一方面利用Netfilter^[4]防火墙架构截获TCP/IP协议栈中的数据包,再采用特定的流量过滤、合并算法得到数据流,并将数据流存储于内核缓冲区中;另一方面利用虚拟文件系统技术实现用户空间到内核空间的内存共享机制。在用户空间,一个流量采集daemon程序首先通过系统调用映射到内核缓冲区上,然后直接从内核缓冲区中读取数据流,并发送给集中存储数据的服务器。

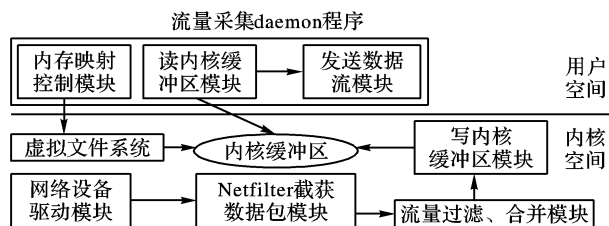


图1 PFC流量采集系统总体结构

2 PFC 网络流量采集系统的实现

PFC网络流量采集系统是基于Linux内核2.6.16设计开发的。内核程序采用标准C语言编写成独立的内核模块,利用Linux内核的模块注册机制在系统启动时加载各模块,在系统结束时卸载各模块,无需对原有Linux内核代码作任何修改。用户空间程序是采用C++语言编写的daemon程序,包含内存映射、读内核缓冲区和发送数据流等功能。

2.1 过滤选项及内核缓冲区的定义

本文所说的过滤选项是按照网络流量信息采集的用途来划分的,实际上是为了PFC系统的应用扩展而设定的,它决定了要采集的数据流的格式。目前PFC定义了三种过滤选项:

```
define NETWORK_ACCOUNTING /* 用于网络计费 */
define NETWORK_PASSIVE_MEASUREMENT /* 用于网络被动测量 */
define NETWORK_MANAGEMENT /* 用于网络管理 */
```

物理上,内核缓冲区是当PFC系统启动时在Linux内核内存空间分配的一块连续页面的静态独占内存,用来存储对截获的数据包进行过滤、合并后的数据流,其大小根据流量过滤选项的不同而不同,不过为了便于内存管理,要保证是Linux内存页面大小的整倍数。在Linux内核2.6.16中,内存页面大小为4kB,所以内核缓冲区的大小为4kB的整倍数。逻辑上,内核缓冲区是由许多数据记录构成的一个大数组,每条数据记录就是一个数据流,而数据流的结构由流量过滤选项来定。另外内核缓冲区还有一个头部(Header),存储对数据流和数据包的统计信息。图2和图3分别是内核缓冲区定义和数据流的一般定义。数据流由一个Header和若干个Packet data构成,Header包含序列号、packet个数等信息。

从TCP/IP协议栈中截获的数据包一般不需要全部采集,而是根据过滤选项来确定哪些信息需要采集,从而也决定着数据流中Packet data的具体数据格式。例如过滤选项为NETWORK_ACCOUNTING时,由于源IP、目的IP和目的端口相同的Packet data可以合并成一个Packet data,所以一个数据流只需要一个Header和一个Packet data,其定义如下:

```
struct packet_data
{
    u_int32_t saddr; /* 源 IP */
    u_int32_t daddr; /* 目的 IP */
    u_int16_t dport; /* 目的端口 */
    u_int32_t in_len; /* 下行流量 */
    u_int32_t out_len; /* 上行流量 */
    char mark; /* 用于访问控制信息的标识 */
} __attribute__((packed));
```

该Packet data的大小为19 Byte,加上Packet flow的Header大小9 Byte,则整个Packet flow的大小为28 Byte。这时如果内核缓冲区大小设定为512×4kB,即2MB,那么,去掉缓冲区的头部占12 Byte外,整个内核缓冲区最多可以容纳74897个Packet flow。因此,即使当网络负载到达峰值时,用户空间程序仍有足够的缓冲时间从内核读取各个Packet flow。

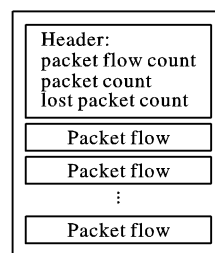


图2 内核缓冲区的定义

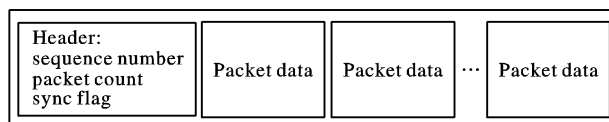


图3 数据流的一般定义

2.2 数据包的截获

在PFC流量采集系统中,数据包的截获是利用Netfilter防火墙架构来实现的。Netfilter是Linux内核2.4.x和2.6.x系列中可选安装的一套功能强大而灵活的防火墙架构,它的前身是Linux内核2.0.x中的ipfwadm系统和内核2.2.x中的ipchains系统^[4]。Netfilter在Linux内核中的IPv4、IPv6和DECnet等网络协议栈中都有相应的实现^[4],PFC系统是基于应用最广泛的IPv4协议栈上实现的Netfilter。IPv4协议栈为了支持Netfilter,在数据包的游历路线中选择了5个参考点,允许内核开发者在这5个参考点注册回调函数,称作钩子(HOOK)。这5个参考点分别是:PREROUTING、LOCAL-IN、FORWARD、LOCAL-OUT和POSTROUTING。PFC系统选择FORWARD参考点注册回调函数,并设置其优先级最低,从而保证只有那些已经通过所有防火墙过滤规则将要转发的数据包才被截获。

回调函数的注册与注销类似于下面的代码:

```
static struct nf_hook_ops ipt_ops[] = { { { NULL, NULL }, ipt_
hook_getpacket, PF_INET, NF_IP_FORWARD, NF_IP_PRI_
LAST }, }, nf_register_hook(&ipt_ops[0]);
...
nf_unregister_hook(&ipt_ops[0]);
```

在以上代码中ipt_hook_getpacket是用来截获数据包的回调函数,该函数有一个重要的输入参数struct sk_buff * *pskb,而sk_buff是Linux内核中贯穿整个TCP/IP协议栈的存储数据包的数据结构,从而能够从sk_buff中获得需要的任何流量信息,暂存到Packet data的数据结构中。

2.3 流量信息的过滤、合并

同传统的基于包过滤的流量信息采集系统不同,PFC系统将流量信息的过滤、合并放到内核空间来处理,这样会减少

用户空间 daemon 程序读取内核缓冲区的次数,降低系统开销。

这里流量信息的过滤并不是依据防火墙规则针对整个数据包进行取舍,而是指根据不同需求确定将要采集数据包内的哪些内容作为一个 Packet data。如 2.1 节所述,PFC 系统设置了不同的过滤选项,事实上这些过滤选项是作为编译宏定义,静态地决定着 Packet data 的格式以及截获数据包的回调函数的实现。

对流量信息的合并是指根据某种过滤选项将截获到的多个 Packet data 合并成一个 Packet flow。例如,过滤选项为 NETWORK_ACCOUNTING 时,可以将一段时间内具有相同源 IP 地址、目的地址和目的端口号的 Packet data 只用一个

Packet data 来表示,而这个 Packet data 的上行流量/下行流量是原来那些 Packet data 的上行流量/下行流量之和,再加上 Header 信息,就得到一个占用内存空间很小的 Packet flow,最后存储到内核缓冲区。流量信息合并的目标是减少 Packet data 的大小和个数,在 PFC 中是借助哈希表来实现的。如图 4 所示,针对所截获的每一个 Packet data,先经过哈希计算找到哈希表中对应元素。哈希表的每个元素是一个存储各个 Packet flow 在内核缓冲区中的偏移量的链表的头节点。因此找到哈希表的元素后,遍历以该元素为头节点的链表,取出各个链表节点所指向的 Packet flow,依据过滤选项逐一匹配,将该 Packet data 合并到某个已经存在的 Packet flow,或者新建一个 Packet flow。

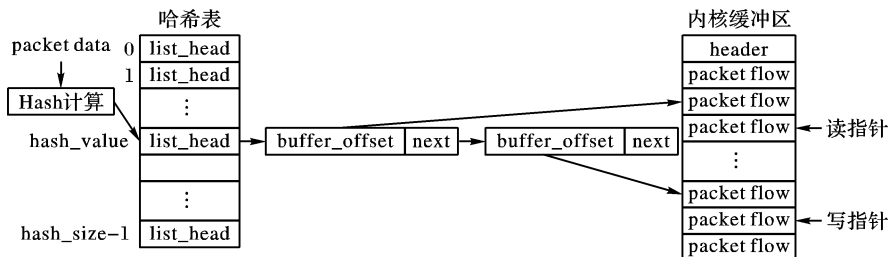


图4 流量信息合并示意图

2.4 用户空间到内核空间的内存映射

共享内存几乎是每种操作系统都会提供的一种进程间的通信机制,但是传统的共享内存机制只支持用户空间的内存共享,对于内核空间的某一块内存,用户空间是无法访问的。然而,PFC 流量采集系统想要突破用户空间和内核空间内存拷贝的瓶颈,必须抛弃传统实现上利用 IO 调用的方法,而需要用户空间的 daemon 程序直接访问内核缓冲区。为此,在 PFC 系统中,特别采用虚拟文件系统技术开发了一个 misc 字符设备驱动模块^[2] pfcmap,在 PFC 系统启动时自动加载,为用户空间 daemon 程序访问内核缓冲区提供内存映射支持,如图 5 所示,具体实现步骤如下:

1)在/dev 目录下创建一个字符设备文件“pfcmap”,主次设备号分别为 10 和 63;

2)开发针对特殊设备“pfcmap”的 misc 设备驱动模块 pfcmap,设备驱动模块是内核程序,能够直接操作内核态内存。pfcmap 的设备文件描述如下:

```
static struct miscdevice pfcmap_device =
{
    minor: MISC_DYNAMIC_MINOR,          /* 次设备号 */
    name: "pfcmap",                      /* 设备文件名称 */
    fops: & pfcmap_fops                  /* 设备文件操作结构 */
};
```

其中 pfcmap_fops 的定义为:

```
static struct file_operations pfcmap_fops =
{
    mmap: pfcmap_mmap,
    /* 定义系统调用 mmap 的实现函数,在 pfcmap_mmap 函数中实现内存映射 */
    open: pfcmap_open,
    /* 定义系统调用 open 的实现函数,目前为空操作 */
    release: pfcmap_close,
    /* 定义系统调用 release 的实现函数,目前为空操作 */
};
```

其中 pfcmap_mmap 函数是 mmap 系统调用的在内核中的实现函数,具体实现了将用户空间进程请求的虚拟内存(用 struct vm_area 表示)映射到内核缓冲区,其核心是调用了 Linux 核心态内存页面映射函数 remap_page_range。

3)在 pfcmap 模块加载时注册设备文件“pfcmap”,调用

Linux 内核函数 misc_register(&pfcmap_device),在 pfcmap 模块卸载时注销设备文件“pfcmap”,调用 Linux 内核函数 misc_unregister(&pfcmap_device)。

4)用户空间 daemon 程序 open 设备文件“pfcmap”,然后对“pfcmap”执行系统调用 mmap,即调用内核驱动模块中的 pfcmap_mmap 函数,将用户空间内存指针映射到内核缓冲区。

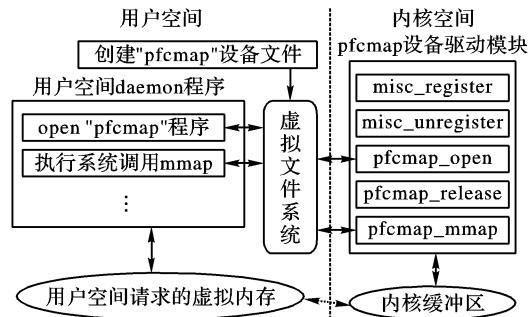


图5 用户空间到内核空间的内存映射

2.5 内核缓冲区的读写同步

内核缓冲区是 PFC 系统的核心数据结构,流量信息过滤、合并模块向内核缓冲区中不断写入 Packet flow,用户空间 daemon 程序从内核缓冲区中不断读出 Packet flow,因此分别有一个写指针和一个读指针同时操作内核缓冲区,这就存在一个读写同步问题。即必须要保证写指针当前所指向的 Packet flow 是空的或已经被读取,读指针当前所指向的 Packet flow 有实际数据并且尚未被读取,而且还要保证读指针和写指针不能同时操作同一个 Packet flow。

在 PFC 系统中,对内核缓冲区的读写同步是通过在每个 Packet flow 的头部设置一个同步标识(sync flag)实现的。这个 sync flag 用一个字节来表示,取其第 0 位指示该 Packet flow 是否被锁定,即是否正在被读或写,取其第 1 位指示该 Packet flow 是否包含有效数据。这样,对于写指针,只有当 sync flag 的第 0 位和第 1 位的值都为 0 时,才可以向该 Packet flow 写入数据,并且在开始写数据之前将 sync flag 的第 0 位的值设置为 1,在写完数据后将 sync flag 的第 1 位的值设置为 1,同时将 sync flag 的第 0 位的值重置为 0。对于读指针,只有当

sync flag 的第0位的值为0,第1位的值为1时,才可以从该 Packet flow 读取数据,并且在开始读取数据之前将 sync flag 的第0位的值设置为1,在读取完数据后将 sync flag 的第1位的值设置为0,同时将 sync flag 的第0位的值重置为0。

3 系统性能与准确性测试

为了验证 PFC 网络流量采集系统的适用性,需要对其性能和准确性进行测试。测试环境的软硬件配置如下:

1) 网关设备:一台高性能微机工作站,硬件配置为4个 Intel Pentium 4 2.80 GHz CPU,16 GB 内存,70 GB SCSI 硬盘, Intel PRO/1000T Desktop Adapter 千兆网卡2块;

2) 操作系统:SUSE Linux Enterprise Desktop 10, Linux 内核 2.6.16;

3) 网络发包设备:SmartBits 6000C 数据网络测试平台;

4) 其他微机工作站:至少2台,分别用于 SmartBits 管理客户端和存储数据的服务器;

5) 其他网络设备:交换机、路由器各一台,双绞线若干。

3.1 性能测试

性能测试的目标是要测试 PFC 系统对承载其运行的整个网关设备的性能的影响。为了实现这一目标就要针对某一特定网关设备分别测试其未加载 PFC 系统时的性能和加载 PFC 系统后的性能,并比较二者之间的差异。网关设备的性能一般用吞吐量和 CPU 负载这两个参数来衡量。为此,定义下面两个性能参数予以测量:

1) PFC 系统的相对吞吐量::=((未加载 PFC 的吞吐量 - 加载 PFC 的吞吐量)/未加载 PFC 的吞吐量) × 100%;

2) PFC 系统的相对 CPU 负载::=((未加载 PFC 的 CPU 负载 - 加载 PFC 的 CPU 负载)/未加载 PFC 的 CPU 负载) × 100%。

基于本节定义的测试环境,利用 SmartBits 发包器向网关设备发送包长度为 1518 Byte 的 UDP 数据包,在限制不同网络带宽情况下测得系统的吞吐量和 CPU 负载,然后分别计算出 PFC 系统的相对吞吐量和相对 CPU 负载,图 6 为测试结果。从图 6 中可以看出,PFC 系统的相对吞吐量和相对 CPU 负载在不同网络带宽下虽然略有起伏,但是都在 3% 以下,因此可以说 PFC 系统对网关设备的性能影响是可以接受的。

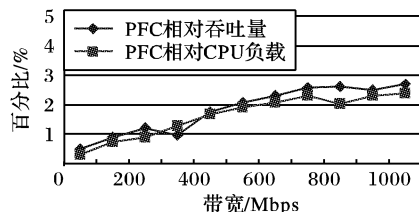


图6 PFC系统对网关设备性能的影响

3.2 准确性测试

准确性测试的目标是测试 PFC 系统对网关设备数据包转发的影响,即测试 PFC 系统的丢包率。需要指出的是,影响整个网关丢包率的因素很多,在 TCP/IP 协议栈的各个层次都可能造成丢包,这里只研究 PFC 系统所造成的丢包率。具体地,PFC 系统的丢包率是指那些被网关设备转发的但是由于内核缓冲区溢出(即 Packet flow 没有被用户空间 daemon 程序及时读取)而没有被采集到的数据包个数占有被转发数据包个数的百分比。在 PFC 系统运行过程中,没有被采集到的数据包个数会被流量合并模块及时更新到内核缓冲区头部

的 lost packet count 域中。用户空间 daemon 程序可以在需要的时候读取 lost packet count 域的值,用来计算丢包率。

基于本节定义的测试环境,在限制不同网络带宽情况下,测得 PFC 系统的丢包率。同时为了跟传统上通过 IO 中断调用从内核采集网络流量的方法作对比,特别的基于常用的 Libpcap 库编写了 Libpcap 流量采集程序,并对 Libpcap 程序的丢包率也作了测试。测试结果如图 7 所示。可以看出在网络负载小于 600MB/s 时,PFC 系统的丢包率一直为 0,在网络负载大于 600MB/s 后,PFC 系统的丢包率才渐渐增加,并且最高在 1% 多一点。而相对于 Libpcap 程序,在网络负载大于 400MB/s 后,PFC 系统的丢包率有显著的降低。这说明,PFC 系统的流量采集的准确性是可以接受的,并且相对于传统流量采集系统有所改进。

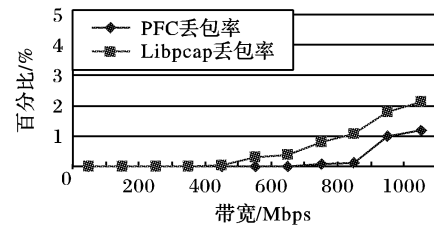


图7 PFC系统的准确性测试结果

4 结语

在分析了当前基于 Linux 平台网络流量采集系统的优缺点和传统上基于在线数据包过滤流量采集系统的性能瓶颈问题的基础上,设计并实现了一种 Linux 平台上基于包过滤的网络流量采集系统 PFC。PFC 系统将流量信息过滤、合并功能转移到内核空间去处理,并且用户空间程序采用共享内存的机制读取内核缓冲区,从而大大减少用户空间和内核空间之间的内存拷贝次数,降低了系统开销。实验证明,PFC 系统在系统性能和流量采集的准确性方面都表现良好,并且相对于传统流量采集系统有较大改进。PFC 系统已经在相关的网络认证计费产品中得到应用,取得了一定的社会和经济效益。

参考文献:

- [1] BROWNEE N. Network management and real time traffic flow measurement[J]. Journal of Network and Systems Management, 1998, 6(2): 223-227
- [2] RUBINI A. Linux device drivers[M]. 2nd ed. America: O'Reilly Press, 2001.
- [3] Spirent Communications, Inc. White paper: SmartBits 6000C product overview[DB/OL]. [2007-02-01]. <http://www.spirent.com/documents/1050.pdf?wt=2&az=c=dc>.
- [4] Netfilter.org. Netfilter firewalling, NAT, and packet mangling for Linux[DB/OL]. [2007-02-04]. <http://www.netfilter.org/>.
- [5] AdventNet, Inc. Network traffic analysis with netflow[DB/OL]. [2007-02-07]. <http://manageengine.adventnet.com/products/netflow/netflow-traffic-analysis.html>.
- [6] LI W Y, WU D, ZHOU B. A study of traffic collection techniques for network management and accounting systems[C]// Proceedings of the 8th International Conference on Computer Supported Cooperative Work in Design 2004. [S.l.]: IEEE Press, 2004, 1: 578-581.
- [7] IETF.org. A Simple Network Management Protocol (SNMP)[DB/OL]. [2007-02-05]. <http://www.ietf.org/rfc/rfc1157.txt>.
- [8] SourceForge.net. Libnet/Libpcap project overview[DB/OL]. [2007-02-04]. <http://sourceforge.net/projects/libpcap/>, and <http://libnet.sourceforge.net/>.