第33卷 第12期 计 算 机 工 程 2007年6月
Vol.33 No.12 Computer Engineering June 2007
·软件技术与数据库· 文章编号：1000—3428(2007)12—0071—02 文献标识码：A 中图分类号：TP393.07

基于Linux内核Netfilter框架的P2P管理

李 健[1,2]，王 玲[1]，李 俊[1]，阎保平[1]
（1. 中国科学院计算机网络信息中心，北京 100080；2. 中国科学院研究生院，北京 100080）

摘 要：在介绍P2P技术和Linux内核Netfilter框架的基础上，提出了识别P2P网络数据包的方法——端口识别法和特征码识别法。介绍了特征码识别法，讨论了如何获取P2P特征码并列出部分已知特征码，阐述了如何利用Netfilter框架进行P2P识别与管理，并进行了简单分析与总结。

关键词：P2P管理；Netfilter；P2P特征码识别

Management of P2P Based on Netfilter Framework of Linux Kernel

LI Jian[1,2], WANG Ling[1], LI Jun[1], YAN Baoping[1]
(1. Computer Network Information Center, Chinese Academy of Sciences, Beijing 100080;
2. Graduate School, Chinese Academy of Sciences, Beijing 100080)

【Abstract】Based on introduction to the P2P technology and Linux kernel Netfilter framework, this paper brings forward the methods, namely, port recognition method and condition code recognition method to identify P2P network packets. It introduces condition code recognition method and discusses how to get P2P condition codes and lists some known condition codes. It also introduces how to identify and manage P2P by Netfilter framework, and gives a brief analysis and summary.

【Key words】P2P management; Netfilter; P2P condition code identification

计算机对等互联网（Peer-to-Peer Network，P2P）技术是当前国际计算机网络技术研究领域的一个热点。P2P技术适用于在不同PC用户之间不经过中继设备直接交换数据或服务，极大地提高了互联网中信息、带宽和计算资源的利用率。但是P2P技术在带来众多益处的同时也存在许多问题。诸如它有较为严重的安全隐患，同时容易导致网络带宽的不合理分配，尤其在被用来进行大量数据下载的情况下，甚至会引起网络拥塞，降低其他业务的性能。这给企业整体网络应用带来巨大压力，企业正常业务网的带宽也会受到影响。P2P管理技术因而成为日益关注的问题。

本文通过分析P2P技术，提出了一种识别P2P网络数据包的方法，并利用Linux内核Netfilter框架简单实现了P2P管理。

1 背景知识

1.1 计算机对等互联网

P2P技术是通过直接交换共享计算机资源与服务。IBM对P2P的定义是：P2P系统由若干互联协作的计算机构成，且至少具有如下特征之一：系统依存于边缘化(非中央式服务器)设备的主动协作，每个成员直接从其他成员而非服务器的参与中受益；系统中成员同时扮演服务器与客户端的角色；系统应用的用户能够意识到彼此的存在，构成一个虚拟或实际的群体[1]。P2P技术打破了传统的客户机/服务器模式，其中每个结点的地位都是相同的，每个结点既充当服务器，为其他结点提供服务，又充当客户机，享用其他结点提供的服务。P2P技术将推动互联网的存储模式由现在的“内容位于中心”模式转向“内容位于边缘（用户）”模式。P2P改变了互联网中以大网站为中心的状态，将权利交还给用户，并重返“非中心化”[2]。

P2P技术在对等计算、协同工作等方面具有优势，现在已经有很多商业化产品。但因为P2P技术也存在很多负面影响，例如安全性较差、网络资源滥用以及大量垃圾信息等问题，所以应该加强对P2P应用的监控与管理。

1.2 Netfilter框架

随着Linux内核版本的不断演变，先后出现了3种内核防火墙框架。最初为ipfwadm框架，随后产生ipchains框架，在此基础上经过完善形成了Netfilter框架。Netfilter框架并不是防火墙的具体实现，而是用于扩展网络服务的结构化底层框架。为了利用该结构，Linux在IPv4协议栈中定义了5个挂载点，以便对网络数据包进行自定义操作。挂载点的位置如图1所示。

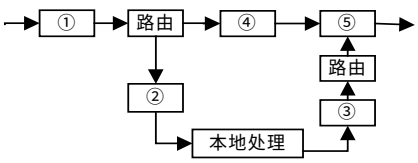

图1 IPv4中挂载点的位置

各挂载点的简单说明如表1所示。

作者简介：李 健(1979－)，男，博士生，主研方向：网络安全，网络管理，下一代互联网技术；王 玲，博士生；李 俊、阎保平，研究员

收稿日期：2006-06-30 **E-mail：**lijian@cstnet.cn

表 1 Netfilter 框架挂载点说明

序号	挂载点	作用对象	实现功能
1	NF_IP_PRE_ROUTING	刚进入网络层的 IP 数据包	源地址转换(SNAT)
2	NF_IP_LOCAL_IN	发往本机的 IP 数据包	过滤输入包
3	NF_IP_LOCAL_OUT	本地发出的数据包	过滤输出包
4	NF_IP_FORWARD	需要转发的 IP 数据包	过滤转发包
5	NF_IP_POST_ROUTING	从设备发出的所有数据包	目的地址转换(DNAT)

通过Netfilter框架提供的API，即可在挂载点定义自己的函数来控制网络数据包[3]。

2 P2P 协议的识别

要管理 P2P，首先必须能识别 P2P 网络流量。随着 BitTorrent 程序源代码的公开，P2P 应用软件越来越多。如何准确快速地识别 P2P 网络数据包成为比较困难的问题。一种比较简单的方法是直接通过已知 P2P 软件常用端口判定 P2P 网络流量，但这可能导致使用其中某端口的其他应用无法正常使用。另外如果 P2P 应用软件能够动态调整端口，则该方法就无能为力了。例如 BitTorrent 通常在 6881 端口监听，若该端口被占用，则一直尝试到 6889 端口。本文采用归纳各种 P2P 应用软件的网络数据包特征码的方法加以判定。可以通过如下 2 种途径获取特征码：

(1)参考相关 P2P 应用软件规范说明书

如BitTorrent[4]、eDonkey和eMule[5]等软件都有详细的规范说明书，分析其说明书就可以归纳出特征码。例如，通过规范说明书，可以获得部分UDP类型的P2P数据包特征码，如表 2 所示。

表 2 部分 UDP 类型 P2P 数据包特征

应用程序	协议	类型	IP 负载大小
eDonkey	0xe3	0x9a	26
		0x96	14
eMule	0xc5	0x91	12
		0x92	10
		0x93	10
Kad	0xe4	0x50	12
		0x59	10

类似地，通过学习 BitTorrent 协议规范说明书，可以分析出 BitTorrent 特征码。BitTorrent 所使用的对等协议实际由一个握手开始。握手数据包格式为<pstrlen><pstr><reserved><info_hash><peer_id>。其中 pstrlen 值为 19，表示 pstr 所代表的协议标识符“ BitTorrent protocol ”的长度。8B 保留字段 reserved 当前都置 0。对元文件中 info 信息进行 SHA1 运算后的哈希值长度为 20B。而 20B 的 peer_id 则为客户端的唯一标识符。接收方也会对元文件中 info 信息进行哈希运算。如果运算结果与 info_hash 不同则表示双方想传输的不是同一文件，所以中断连接；否则在握手之后就是循环的消息流，且每个消息的前面都有一个数字表示消息长度。由此可以看出，将“ BitTorrent protocol ”作为 BitTorrent 数据包的特征码，结合 TCP 协议的特点就可以判断出 BitTorrent 协议数据包。

(2)通过捕获 P2P 网络数据包分析对应特征码

有些 P2P 应用软件虽然较为流行，但并不提供相关的规范说明书。分析并寻找特征码的过程会比较困难，对此可以通过 sniffer 程序捕获 P2P 应用软件的网络数据包，再进行手工分析，从中找出特征码。本文利用基于 Netfilter 框架自行开发的程序捕获网络数据包，分析出了部分 P2P 应用软件的特征码。例如，通过分析 BitTorrent 网络数据包完全可以得到特征码“ BitTorrent protocol ”。这也与 BitTorrent 协议规范说明书中的相关内容相互认证。类似地，可以找到其他 P2P 软件网络数据包的特征码。部分特征码如表 3 所示。

表 3 部分 P2P 软件网络数据包特征码

P2P 软件名称	特征码
BitTorrent	BitTorrent protocol
卡盟(KAMUN)	KamunPeers protocol
百度下吧	BaiduP2P
Gnutella	GND 或者 GNUTELLA
KaZaA	KaZaA
Ares Galaxy	PUSH SHA1:

虽然国内 P2P 应用软件种类繁多，但是基本原理却相差无几，大都可以分析得到相应的特征码，此处不再赘述。

3 利用 Netfilter 框架实现的 P2P 管理

本系统利用 Linux 内核 Netfilter 框架实现对 P2P 网络流量的简单管理。为了便于添加新特征码及调整控制策略，本系统将特征码及其控制策略都写入配置文件中，从而与具体的实现代码分离，可在不修改源代码的情况下进行升级。配置文件简单示例如下：

```
BitTorrent  “BitTorrent protocol”  -1
```

其中，第 1 列为 P2P 软件名；第 2 列为特征码；第 3 列为空，以便于添加新特征码；最后 1 列为控制策略，取值为-1 表示继续传输，为 0 则丢弃，其他正数则表示该类型数据包的最高流速（限速，单位为 KB）。UDP 部分与此类似，可参考表 2，此处不再赘述。

在内核 2.4.20 中，利用 Netfilter 框架必须用如下函数向内核注册自定义处理函数：

```
int nf_register_hook(struct nf_hook_ops *reg)
```

注销函数原型如下：

```
void nf_unregister_hook(struct nf_hook_ops *reg)
```

其中参数类型定义如下：

```
struct nf_hook_ops{ struct list_head list;
                    nf_hookfn * hook;
                    int pf;
                    int hooknum;
                    int priority;};
```

该结构中，list 一般初始化为{NULL,NULL}；pf 则为 PF_INET；hooknum 是选择的挂载点，一个挂载点可以有多个处理函数，其执行顺序通过 priority 值决定。用户可以自定义 hook 函数，该类型定义如下：

```
unsigned int nf_hookfn(unsigned int hooknum, struct sk_buff ** skb , const struct net_device* in , const struct net_device* out, int (*okfn)(struct sk_buff *));
```

该函数返回值可为 NF_ACCEPT（内核继续传输数据包）、NF_DROP（内核丢弃数据包）或者 NF_STOLEN（模块接管该数据包，不再继续传输该数据包）、NF_QUEUE（对该数据包进行排队，通常用于将数据包给用户空间的进程进行处理）和 NF_REPEAT（重复调用该 hook 函数）等。这里，主要的工作就是定义 hook 函数，在该函数中过滤网络数据包，部分关键代码如下：

```
static unsigned int p2pfilter(unsigned int hooknum, struct sk_buff ** skb , const struct net_device* in , const struct net_device* out, int (*okfn)(struct sk_buff *))
{   struct   sk_buff * sb=*skb;
  if(sb->nh.iph->protocol==IPPROTO_TCP)
  p2p_tcp(sb);     //识别并处理 TCP 类型的 P2P 数据包
  else if(sb->nh.iph->protocol==IPPROTO_UDP)
  p2p_udp(sb);     //识别并处理 UDP 类型 P2P 数据包
  return NF_ACCEPT;   }
```

(下转第 75 页)