第3章 基于通信网络拓扑结构的P2P流量识别模型(P2P-CNTIM)

Internet 地址结构划分导致了整个 Internet 网络结构的变化，这种变化对于 C/S 架构的网络应用来说，不会受到影响，但是 P2P 应用需要采用各种穿越技术，使得 NAT 后的内网主机也具有担当服务器的能力，因为对于 P2P 应用来说，一个节点需要和其他大量的节点进行交互以完成资源共享和协作计算，这些通信对端节点不仅包含具有公网 IP 地址的服务器，也包含 NAT 后的内网节点[1]。本章分析了 P2P 在 Internet 上的通信网络拓扑特征后，给出一个基于通信网络拓扑结构的 P2P 流量识别模型(P2P-CNTIM)[2,30]。该模型根据 P2P 独有的通信网络拓扑结构特征，使用多主机特征以及通信对端类型特征对 P2P 流量进行识别，并将这两个特征有机地结合起来以提高识别的准确率和识别效率[2,18,26]。

3.1 基于通信网络拓扑结构的 P2P 流量识别模型(P2P-CNTIM)概述

3.1.1 P2P 通信网络拓扑特征分析

在 Internet 环境下，如果网络应用是 C/S 结构，客户端可以是处在 NAT 后的主机或者是具有公网 IP 的主机，但服务器一定是具有公网 IP 的主机。如果该网络应用是 P2P 结构，从概率上来讲，和其通信的主机有一部分是处于 NAT 后的内网主机[3]。由于 P2P 应用需要采用各种技术来使得处于 NAT 后的主机之间能够建立通信，同时由于 P2P 技术的特点，一台主机在一段时间内需要和多台主机进行通信[4]。在这个过程中，从某台主机发送和接收的数据包的 IP 地址来看，这些数据包的最终目标地址构成的网络拓扑结构如图 3-1 所示[5,6]。

图 3-1 是在运行 P2P 应用的被观察节点和其他节点进行通信的网络拓扑图。根据 P2P 技术特点和 Internet 的地址结构特点，可以得到运行 P2P 应用的节点在网络拓扑结构上的明显特征[7,8,9,10]。

1. 多主机通信特征

在一段时间内，或者在某一个时刻，P2P 节点和多台主机进行数据通信。大部分的 P2P 应用如 BitComet、PPStream、PPLive、QQLive 在作为客户端下载数据的同时，也作为服务端在向其他节点提供数据，同时 P2P 应用一般都需要从多个服务器上下载数据，所以运行

P2P 应用的节点会和多台主机进行数据通信。

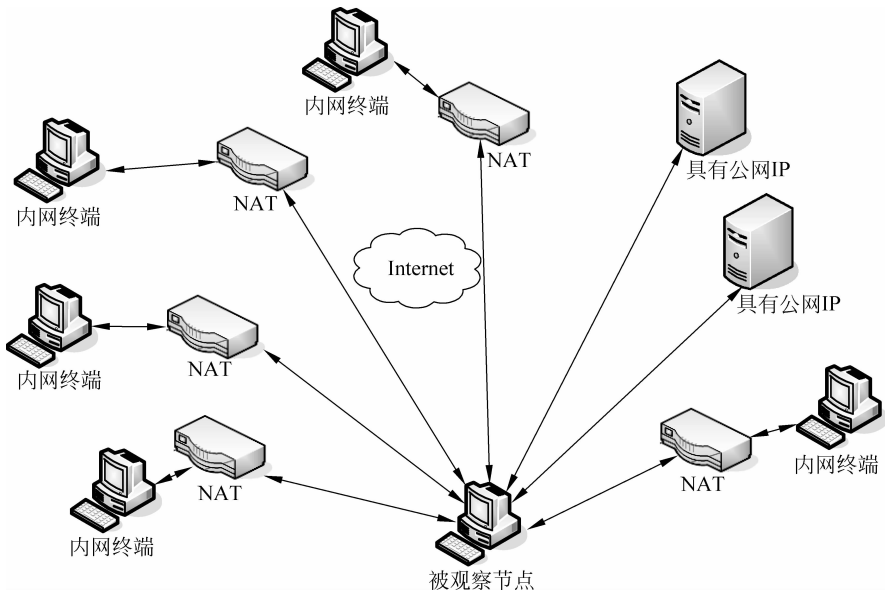


图 3-1　运行 P2P 应用的节点的通信网络拓扑图

2. 通信对端类型特征(和 NAT 后内网主机通信特征)

P2P 应用为了发挥各个节点的数据共享性,会尽可能采用 P2P 穿越技术使得内网节点也能够担当服务器的角色,否则,如果仅仅使公网上的节点担当服务器,那么该 P2P 应用的共享度必然会受到很大的影响,因为在 Internet 上拥有公网 IP 的终端毕竟只占很少的一部分,所以在一般的情况下,和被观察节点进行通信的节点有一部分是 NAT 后的内网节点,由于 P2P 应用的对端主机即包含 NAT 后内网主机,也包含具有独立公网 IP 的主机,因此在本文中该特征称为通信对端类型特征。

3. 具有公网 IP 主机通信特征

P2P 节点进行通信的一部分主机是具有公网 IP 的主机,P2P 应用首先连接公网上的服务器去查询下载目标的信息,需要从服务器上去获取拥有资源的其他节点信息,需要公网上的服务器充当 NAT 穿越的中介,所以和被观察节点进行通信的节点有一部分是具有公网 IP 的节点。

上面总结了 P2P 节点进行通信的网络拓扑特征,除了这些特征之外,根据已有的工作,P2P 流还有以下明显的统计特征。

(1) 交互数据量特征。P2P 节点和多台主机之间交换的数据量比较大,由于大部分流行的 P2P 应用是属于共享文件类别的,因此在 P2P 软件运行时,就会产生大量的网络流量。又由于 P2P 节点有着客户端和服务端的双重角色,因此其上行流量在其产生的总流量中占有相当的比例。

(2) 数据包长度(Packet Length)特征。即 P2P 软件运行时所发送的数据包长度。这种数据包长度对同一种 P2P 软件具有普遍性,而相对于其他 P2P 软件具有特殊性。

(3) 典型 P2P 应用端口(P2P Ports)特征。通过人们对各种已有 P2P 软件的研究,总结出了很多热门 P2P 软件的通用端口,如表 3-1 所示。这些端口中有的是软件开发商默认使用的。

表 3-1　部分 P2P 软件使用的协议端口

序号	软件名称	协议端口
1	BT	TCP: 6881～6889
2	eMule	TCP: 4661～4662
3	Gnutella	TCP: 6364
4	DirectConnect	TCP: 411～417
5	FastTrack	TCP: 1214
6	POCO	TCP: 9000,5354 UDP: 5356
7	Netfairy	TCP: 7777,7778,11300
8	酷狗	TCP: 7000,3318
9	PPLive 网络电视	UDP: 4004 TCP: 8008
10	网酷	TCP: 2122
11	Winmx	TCP: 5690
12	iMesh 5	TCP: 4662
13	iLink 1.1	TCP: 5000
14	石头(Openext)	TCP: 5467,2500,4173,10002,10003
15	酷乐	TCP: 6801,6800,7003
16	eDonkey2000	TCP: 4371,4662
17	比特精灵	TCP: 16881
18	QQ 直播	UDP: 13000～14000
	…	…

(4) 通信协议特征。传统的 C/S 架构应用的通信过程中,一般只采用一个协议(TCP 或 UDP),但对于 P2P 应用,有一部分软件采用两种协议(TCP 和 UDP)进行通信。

(5) 流平均速度(Average Flow Rate)特征。P2P 应用由于一般是文件的下载,其传输的数据量大,因此其产生的流平均速度也较大。

(6) 流持续时间(Flow Time)特征。有很多 P2P 软件如 BitComet,用户建立下载任务后就可以让其长时间的运行,对应经常使用的下载电影等文件任务来说,一个任务持续的时间比较长。

(7) P2P-TCP 协议的 SYN/ACK 统计特征。

3.1.2　P2P 流量识别确定性特征选择

可以用 P2P 流特征来进行 P2P 流的识别[11],然而有的特征只是针对部分 P2P 应用适用,能够得出符合这些特征的数据流就是某些特定 P2P 流的结论,但是反之,则不能说明不符合这些特征的就不是 P2P 流,如基于端口识别的特征就是根据特定 P2P 应用对应的端口来进行 P2P 流的识别,只要符合这些端口特征,就可以认为对应的 P2P 应用存在,但是不能因为不符合这些特征就判定其不是 P2P 流,这些特征由于不对所有的 P2P 应用适用,使用这些特征来决定一个数据流是不是 P2P 流时,只能把数据流划分为特定应用的 P2P 流或者不是该特定应用的数据流,能够得到 P2P 流的范围较小,所以称为非确定性特征。另一方

面，上述特征中有一些特征能够把一个数据流划为 P2P 流或者非 P2P 流，这些特征是一般 P2P 应用的共有特征，称为确定性特征，如同时和 P2P 节点进行通信的多主机的特征，只要给出阈值，就有理由将数据流划分为 P2P 流或非 P2P 流。

显然非确定性特征只能识别出部分 P2P 流，依赖于非确定性特征来识别有很大的局限性，P2P 识别系统的目标是能够对数据流进行 P2P 和非 P2P 的流分类，然后再对 P2P 流的具体应用进行细分，准确对数据流进行 P2P 流和非 P2P 流的识别是进一步划分到各个具体应用的基础，所以需要找出能够划分数据流为 P2P 流和非 P2P 流的确定性特征。

下面将逐一分析每个 P2P 流的特征是确定性特征还是非确定性特征。

1. 多主机通信特征

对于一个 P2P 应用来说，为了发挥每个节点的共享能力，必然会有多个节点参与到运行任务中，传统的 Internet 网络应用如 Web、FTP 等一般采用固定通信端口，然而 P2P 应用在和不同的目标主机通信时一般采用不同的端口，可以选择一个阈值，在一定的时间范围内，当一个被观察节点通信对端节点数和不同的通信端口数大于某个阈值时，可以认为被观察主机产生了 P2P 流，所以该特征是 P2P 流划分的确定性特征。

2. 通信对端类型特征(和 NAT 后内网主机通信特征)

P2P 应用必须采用 NAT 穿越技术使内网的主机有能力担当服务器的角色，由于多主机通信特征保证了一个节点同时和多个节点在通信，因此可以确定在和被观察主机通信的节点中，如果该应用是 P2P 应用，则有一部分节点是位于 NAT 后的内网节点，当和被观察节点通信的 NAT 后的内网主机的数量大于某个阈值时，可以认为被观察主机产生了 P2P 流，所以该特征也是确定性特征。

3. 和具有公网 IP 主机通信特征

显然，普通的 C/S 应用对应的服务器主机也都是具有公网 IP 的主机，所以该特征不是确定性特征。

4. 交互数据量特征

对于下载/上传数据量的特征，在流量监测的过程中，不可能一直追踪到某一 P2P 软件用户传输完整个共享文件为止，所以一定是以一定时间内的流量总长度为准。这样，问题实质上又转化到流平均速率上。因为一定时间内的流量数据包总长度是和流平均速率成正比的，所以该特征是非确定性特征。

5. 数据包长度(Packet Length)特征

虽然每个 P2P 软件都有其自身的长度特征，但是并不能用这个标准来判断流量的属性。因为在复杂的网络环境下，数据包长度具有很大的相似性和不确定性，所以该特征是非确定性特征。

6. 典型 P2P 应用端口(P2P Ports)特征

由于现在很多 P2P 应用软件都采用了动态端口技术，因此该特征属于非确定性特征。

7. 通信协议特征

有很多 P2P 软件同时采用 TCP 协议和 UDP 传输层协议，但是这并不能作为确定数据流是 P2P 流还是非 P2P 流的充分依据，所以该特征是非确定性特征。

8. 流平均速度(Average Flow Rate)特征

对于流平均速率特征，其实际的分布区间会因网络的不同而不同，如限制 P2P 应用的校园网和企业网与普通的居民小区网络中 P2P 软件运行时的通信速率肯定是不同的；而在同一个网络中，也会因不同时段的网络质量的不同而不同，网络拥塞时的平均速率必然低于网络状况正常情况下的通信速率，所以该特征是非确定性特征。

9. 流持续时间(Flow Time)特征

流持续时间较长是一般 P2P 软件的共有特征，但是其往往和网络带宽等情况相关，所以该特征是非确定性特征。

10. P2P-TCP 协议的 SYN/ACK 统计特征

该统计特征是基于部分 P2P 协议分析而来，其适用的范围只适合于特定的 P2P 应用，所以该特征是非确定性特征。

根据上面的分析结果，得到了用于 P2P 流量识别的统计特征关于确定性和非确定性的分类列表 3-2。可以看出，尽管识别 P2P 流有很多可用的统计特征，然而能够准确将数据流划分为 P2P 流或非 P2P 流的只有多主机特征以及通信对端类型特征。错误地把非 P2P 流量检测为 P2P 流量称为误判，错误地把 P2P 流量检测为非 P2P 流量称为漏判，对于多主机通信特征是确定性的划为还需要进一步增加一些条件，如不同端口数占对端主机数的比例等，以减少误判率。

表 3-2　P2P 流量识别统计特征分类

序号	特 征 名 称	特征分类
1	多主机通信特征	确定性
2	通信对端类型特征(和 NAT 后内网主机通信特征)	确定性
3	和具有公网 IP 主机通信特征	非确定性
4	交互数据量特征	非确定性
5	数据包长度(Packet Length)特征	非确定性
6	典型 P2P 应用端口(P2P Ports)特征	非确定性
7	通信协议特征	非确定性
8	流平均速度(Average Flow Rate)特征	非确定性
9	流持续时间(Flow Time)特征	非确定性
10	P2P-TCP 协议的 SYN/ACK 统计特征	非确定性

通信对端类型特征具有确定性，从 3.1.1 节的论述中可以获知，由于 Internet 的网络地址划分机制以及 P2P 应用的特点，使得 P2P 应用需要采用 NAT 穿透技术使 NAT 后内网主机也具有担当服务器角色的能力，所以和被观察节点通信的对端主机有一部分是位于

NAT 后内网主机，当其超过一定阈值时（该阈值远小于多主机特征的阈值），则我们有理由认为该数据流是 P2P 流。由于 Internet 上绝大部分的网络服务器都是具有公网 IP 的独立主机，因此在一般的情况下，该阈值只要取为 1 就能获得较低的漏检率和误判率，如果考虑到存在很少的一部分 Internet 服务通过 NAT 网关设备来做端口映射，为了降低误判率，可以通过在判断通信对端类型的处理中增加一些限制条件以及适当的增加阈值来达到目的。

另一方面，P2P 流的多主机通信特征的阈值也要随着应用的实际情况进行不断的优化，如果阈值取得过大，则漏判率高而误判率较小，如果阈值取得过小，则误判率高而漏判率下降。

将数据流准确划分为 P2P 流和非 P2P 流是进一步识别 P2P 具体应用的基础，可以单独使用基于多主机通信特征以及基于通信对端类型特征作为 P2P 流量识别唯一依据，然而，基于流量特征的 P2P 流量识别技术具有准确性差、健壮性差的不足，识别 P2P 流的多主机通信特征在设置最优的阈值上存在一定的难度，通信对端类型特征在处理中存在着无法判断所有通信对端类型的不确定因素（约有 42%的通信对端类型可以被确定），实际上这两个特征都反映了 P2P 技术在 Internet 上的独有通信特征，为了获得较小的误判率和漏判率，可以把这两个特征有机地结合起来以获得比使用单个特征更多的选择范围，由于这两个特征都反映了被观察主机的通信网络拓扑结构，因此它们构成了基于通信网络拓扑结构的 P2P 流量识别模型（P2P-CNTIM）主要方法。

3.1.3 获取通信对端类型关键技术

在 3.1.2 节中，通过分析，选择多主机通信特征以及通信对端类型特征作为识别 P2P 流的依据，对于多主机特征的统计处理相对容易，然而对于通信对端类型特征，需要采用可行的技术来识别和被观察主机进行通信的对端类型，即要分辨出通信对端是 NAT 设备后的内网主机还是具有公网 IP 的提供 Internet 服务的主机。

P2P 流量识别通常应用在网关设备附近的位置（为了便于描述，假设该软件运行在网关上），而被观察节点属于内网的主机，和内网主机通信的对端类型可能是具有独立公网 IP 的服务器，也可能是采用了穿透技术的内网主机，分别如图 3-2 和图 3-3 所示。

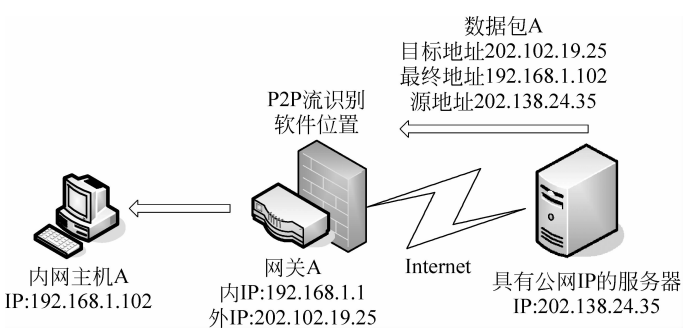


图 3-2 内网主机和具有公网 IP 的服务器通信

在图 3-2 和图 3-3 中，对于网关 A 来说，数据包 A 和数据包 B′来说，其目标地址、最终地址和源地址都是相同的，但是实际上这两个数据包的产生者是不同的，数据包 A 是来自具有公网 IP 地址的服务器的数据包，数据包 B′实际上是来自于内网主机 B 的数据包，内网

主机B发出数据包B到网关B上，网关B对数据包进行了修改，替换了其源IP，然后将数据包经过路由发到了网关A，如果运行在网关A上的P2P流量识别软件能够识别到数据包B′实际上并不是其源地址202.102.24.35产生的，而是其他主机产生的，那么网关A上的P2P流量识别软件也就能识别到和内网主机A通信的对端类型。

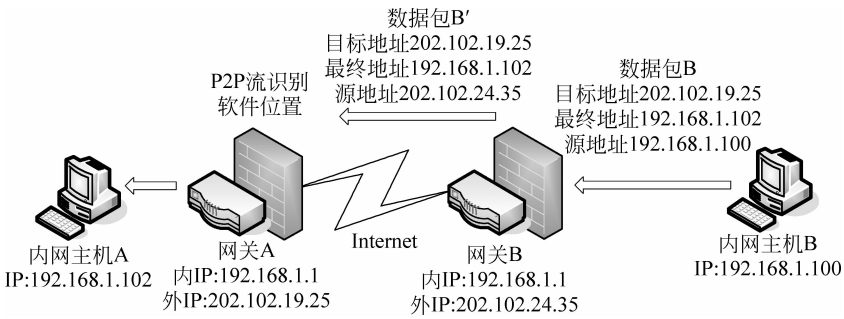


图3-3　内网主机和其他NAT后内网主机通信

在http://www.sflow.org/detectNAT/上，Peter Phaal给出了在交换机上检测内网是否使用NAT设备的技术(sFlow是一种基于标准的最新网络导出协议RFC3176)，该技术主要是根据TTL来计算的。将这一技术原理进一步拓展，以用来解决识别通信对端是否是NAT设备的问题，其原理是根据网络路径长度来计算的，也需要使用TTL作为测试依据。

图3-4给出了IP数据报文格式，在IP报文格式中，用TTL来表示报文的生存时间，TTL(time-to-live)生存时间字段设置了数据报可以经过的最多路由器数。它指定了数据报的生存时间。TTL的初始值由源主机设置(通常为32、64、128、255)，一旦经过一个处理它的路由器，它的值就减去1。当该字段的值为0时，数据报就被丢弃，并发送ICMP报文通知源主机。

图3-4　IP数据报格式及首部中的各字段

在内网主机和具有公网IP的服务器通信中，IP地址为202.138.24.35的服务器发送数据包A到Internet上，该数据包在Internet上每经过跳转一次，其TTL值就减少1，当到了其目标地址网关A时，其减少的TTL值实际上代表了该路径中的跳数(可以用跳数来代表网络距离)，其TTL值减少的值和服务器到网关A之间的实际网络距离是相同的。

当内网主机B发送数据包B给网关B时，网关B修改了数据包B中的源IP地址，并把

其 TTL 值减 1，然后把该数据包发到 Internet 上，该数据包在 Internet 上每经过跳转一次，其 TTL 值就减少 1，一直到目标 IP 地址网关 A 为止，其 TTL 值减少的值为网关 B 到网关 A 之间的实际网络距离再减去 1，也就是对于数据包 B′来说，其 TTL 减少的值和服务器到网关 A 之间的实际网络距离是不相同的，如图 3-5 所示。网关 A 上接收到的相同源 IP 地址 202.102.24.35 的 IP 包中的 TTL 减少值是不同的，数据包 B′中减少的 TTL 为 64－53＝11 跳，而 ping 响应包中 TTL 减少为 64－54＝10，所以可以确定 202.102.24.35 不是数据包 B′的真正产生主机。

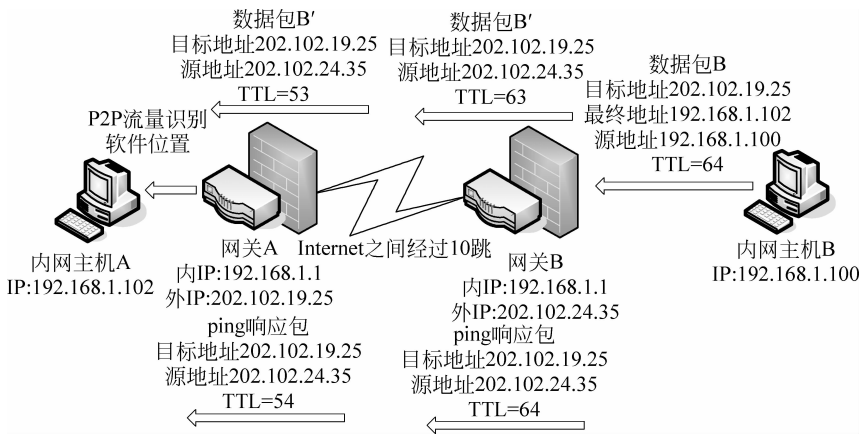


图 3-5　不同类型的通信对端 TTL 区别示意图

根据该原理，如果接收到的数据包中的 TTL 减少值大于接收端与数据包的 IP 地址对应的网络设备（主机）之间的网络距离，就可以认为该数据包的源 IP 地址不是产生该数据包的真正 IP 地址，也就可以认为其通信对端类型是 NAT 后的设备，尽管 TTL 绝对值会由于路由变化而产生波动，然而 Internet 主干网络上的路由设备还是相对稳定的，在很短的一段时间内，两个节点之间通信的数据包中的 TTL 值是一致的。

如果可以获取网关 B 产生的数据包，那么就可以获得网关 A 和网关 B 之间的跳数，可以使用下面的方法。

（1）网关 A 向网关 B 发送 ping 报文，根据 ping 的返回结果中的 TTL 值来计算之间的跳数。

（2）网关 A 向网关 B 发送一大端口的 UDP 报文，网关 B 会返回端口不存在的 ICMP 报文，根据该返回结果中的 TTL 值来计算之间的跳数。

（3）网关 A 向网关 B 发送一个大端口的 TCP 连接报文，网关 B 会返回 TCP 的 RST 报文，根据该返回结果中的 TTL 值来计算之间的跳数。

从 TCP/IP 的协议来看，使用任何一种方法对于对端类型为 NAT 设备来说，得到的结果是一样的，由于现在有一些访问量非常大的 Internet 服务器使用了负载平衡，虚拟 IP 等技术，用上面的 3 种方法可能会得到不一样的结果[12]，另一方面，有很少的一部分 Internet 服务器使用 NAT 多端口映射来使内网的主机具有服务能力，为了降低误判率，对计算两个网关之间的距离的算法增加了下面的两个条件。

（1）只对通信端口大于 10000 的主机进行检测，因为 NAT 设备的映射端口一般都是高

端口的，这样即减少了系统的开销，又避免了对提供 Web、FTP 等普通网络服务的主机的误判；

(2) 同时使用上面的三条规则进行检测，只有检测结果是相同的，对端类型是 NAT 类型才成立，这也是为了降低系统的误判率。

这些条件的增加实际上使得基于通信对端类型特征来进行 P2P 流量识别的阈值可以采用很小的值就能获得较低的误判率和漏检率，图 3-6 给出了识别通信对端类型的伪代码。

```
Recv packet A from hostA and the source port of the packet > 10000
Get the distance DA from packet A by its TTL
Send ping packet to hostA
Send UDP packet with huge dest port to hostA
Send TCP packet with huge dest port to hostA
If(recv ping response packet P and udp response packet U and tcp response packet T)
  Get the distance DP from packet P by its TTL
  Get the distance DU from packet U by its TTL
  Get the distance DT from packet T by its TTL
  if(DP==DU==DT < DA)
     hostA is belonged to the type of NAT
```

图 3-6　通过 ping 测试通信对端类型的伪代码

大量的实验验证了上述算法的可行性和正确性，下面描述了其中的一个实验过程。

图 3-7 为实验网络拓扑图，主机 A、B、C、D 通过 ISP 服务商提供的网关设备接入 Internet，各个网关设备分别获取了一个临时的公网 IP 地址，主机 A、B、C、D 是内网主机，并都有一个内网 IP 地址。Skype 是一个典型的 P2P 应用，为了验证本方法的可行性，在各个主机上分别安装 Skype、QQ、FTP、IE 软件，并向 Skype 注册了 4 个用户，分别为 UserA、UserB、UserC、UserD。

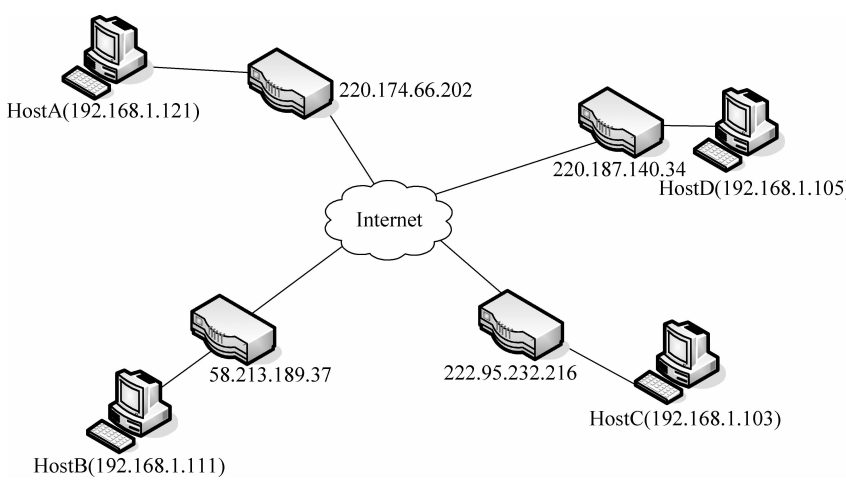


图 3-7　获取通信对端类型实验的网络拓扑图

在实验过程中主机 B 和主机 C 上的用户分别使用 UserB 和 UserC 用户名登录 Skype，在主机 B 上用户向 UserC 建立呼叫，由于 UserC 在线，因此呼叫成功，双方进行会话，在各个主机上的用户同时打开 IE 及 QQ 进行操作使用。下面说明 HostB 上的分析过程。

在 HostB 上分别获取这些网络应有的数据包，如图 3-8 所示。

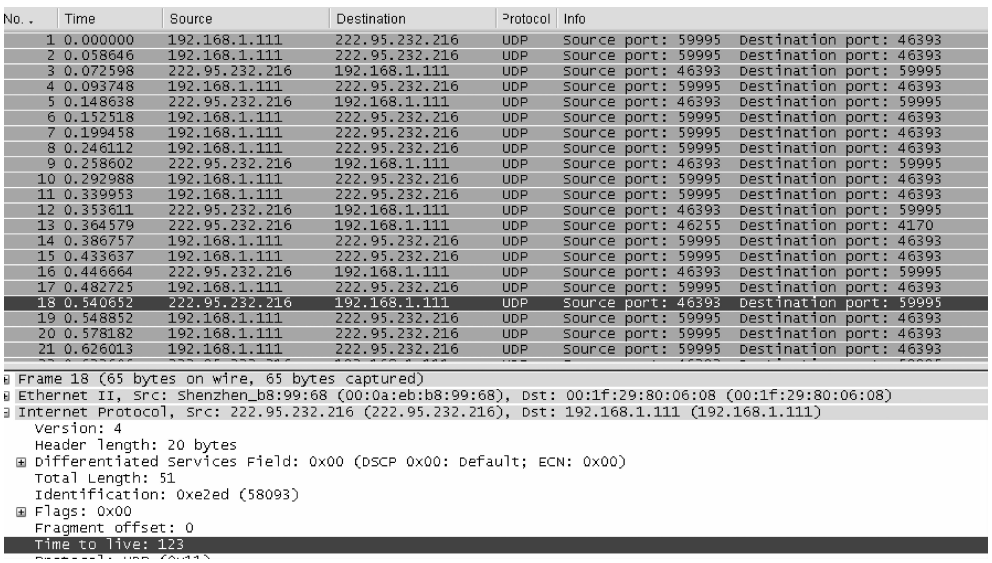


图 3-8　获取通信对端类型实验抓包截图

通过分析可以发现，IP 地址为 222.95.232.216 的主机和 HostB 有大量的数据包交换，且其端口号较高，222.95.232.216 使用的端口是 46393，HostB 端口为 59995，通过端口进程关联可以获取到这些数据包是和 Skype 应用相关的，从 222.95.232.216 发送过来的数据包的 TTL 为 123，由于 TTL 的典型值为 64、128、255，因此路由跳数为 128－123＝5 跳。使用 ping 对方的 IP 地址，获得了图 3-9 的结果。

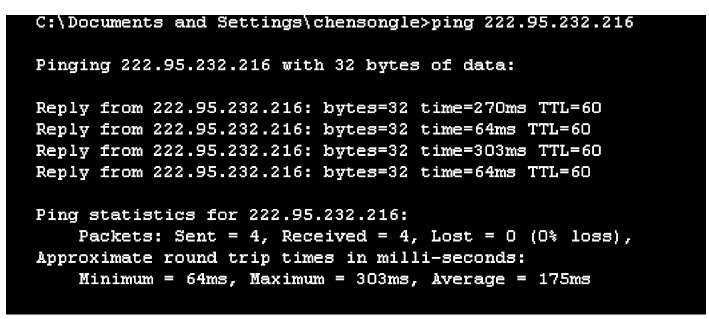


图 3-9　获取通信对端类型实验 ping 返回结果截图

根据 ping 返回的结果，其 TTL＝60，则说明 HostB 和对方 IP 地址之间实际的路由跳数为 64－60＝4 跳，这说明 HostA 上收到的数据包的源 IP 不是 222.95.232.216，222.95.232.216 对应的设备可能是一个 NAT 设备，又向对方发送了一个高端口的 TCP 包和 UDP 包，也得到了相同的结果，所以可以得出结论 222.95.232.216 对应的设备是一个 NAT 设备，这和实际情况一致。

如果所有的网络设备都按照 TCP/IP 协议规定产生 ICMP 报文和 TCP RST 报文，那上面的方法就总是正确的，然而有一部分网络设备为了安全的考虑，关闭了 ICMP 的所有响应报文，在这种情况下，获取通信对端类型是否是 NAT 设备的方法就会失败，根据实验测

试的数据表明,P2P 应用的对端设备中约有 42%可以被判断出,然而我们并不需要获得和被观察节点通信的所有对端主机的类型,在和被观察节点通信的所有对端主机中,只要获取其中的很少一部分通信对端类型为 NAT 后的内网主机,由于识别通信对端类型算法的严格性,因此就可以确定该数据流为 P2P 流。即使在最坏的情况下,由于 P2P-CNTIM 选择了多主机特征以及通信对端类型特征这两个双重特征作为划分数据流的依据,因此还是能够根据多主机的特征来对数据流进行正确划分[13]。

3.2　P2P-CNTIM 流量识别模型中的关键技术

3.2.1　P2P-CNTIM 特征判断函数

1. 多主机特征判断函数定义

P2P 应用软件由于其技术特点一般会和大量的节点进行通信[14],其通信的对端主机数往往大于一个阈值,其产生的不同源端口、对端端口、对端 IP 数很接近,图 3-10 是 BitComet 向其他 Peer 发送连续的连接请求,可以说明这一点。

1696 10.10.80.131	222.80.2.7	2885	24404	TCP	2885 > 24404 [SYN] Seq=0 Len=0
1697 10.10.80.131	58.213.198.83	2886	23664	TCP	2886 > 23664 [SYN] Seq=0 Len=0
1698 10.10.80.131	220.163.52.247	2887	19770	TCP	2887 > 19770 [SYN] Seq=0 Len=0
1699 10.10.80.131	58.209.195.101	2888	21440	TCP	2888 > 21440 [SYN] Seq=0 Len=0
1700 10.10.80.131	202.100.228.98	2889	15854	TCP	2889 > 15854 [SYN] Seq=0 Len=0
1701 10.10.80.131	61.171.35.225	2890	29940	TCP	2890 > 29940 [SYN] Seq=0 Len=0
1702 10.10.80.131	222.68.44.79	2891	27666	TCP	2891 > 27666 [SYN] Seq=0 Len=0
1703 10.10.80.131	218.71.239.67	2892	14061	TCP	2892 > 14061 [SYN] Seq=0 Len=0
1704 10.10.80.131	58.35.12.198	2893	27368	TCP	2893 > 27368 [SYN] Seq=0 Len=0
1705 10.10.80.131	222.71.43.42	2894	23024	TCP	2894 > 23024 [SYN] Seq=0 Len=0
1706 10.10.80.131	24.80.113.24	2895	13299	TCP	2895 > 13299 [SYN] Seq=0 Len=0
1707 10.10.80.131	218.22.136.217	2896	13195	TCP	2896 > 13195 [SYN] Seq=0 Len=0
1708 219.153.5.15	10.10.80.131	20271	2844	TCP	20271 > 2844 [SYN, ACK] Seq=0 A

图 3-10　BitComet 向其他 Peer 发送连续的连接请求

对于 P2P 多主机的连接特征,如果一个主机访问多个不同的 Web 站点,仅仅根据对端 IP 地址数去识别,那么该主机产生的数据流将被错误地判断为 P2P 流。由于 P2P 流量检测的主要目的是控制 P2P 流量,如果检测方法的误判率高,则会导致错误地控制了许多非 P2P 流量,特别是对于采用封堵策略的控制方法,较大的误判率就有可能严重影响非 P2P 用户的网络活动。而对于大部分 P2P 控制策略而言,检测到主要的 P2P 流量已经能够使控制策略有效的发挥作用,所以要尽量降低误判率,为了降低误判率,本文对多主机连接特征增加一些端口限制。

定义 3.1　多主机特征统计变量

dest_ip:被观察主机连接的对端地址。

diff_dest_ip_count:在一段统计时间内不同的对端 IP 地址数。

dest_port:被观察主机一个连接的对端端口。

diff_dest_port_count:被观察主机连接的不同的对端端口数。

src_port:被观察主机连接源端口。

diff_src_port_count:被观察主机的本地源端口总数。

定义 3.2 多主机特征的判断函数

$$\begin{cases} \text{diff\_dest\_ip\_count} > n_1 \\ \text{diff\_dest\_port\_count} > n_2 \\ \text{diff\_src\_port\_count} > n_3 \\ \text{diff\_dest\_port\_count} \approx \text{diff\_dest\_ip\_count} \end{cases} \tag{3-1}$$

式中，n_1、n_2、n_3 为判断阈值，最后一个条件表示通信对端不同端口数和通信对端不同的IP数应该相近。在测试的一段时间内，只要满足上述条件，就可以判断该数据流是P2P流。

2. 通信对端类型特征判断函数定义

定义 3.3 通信对端类型特征统计变量

desp_ip_nat：观察主机的通信对端是NAT后的内网主机。

dest_ip_internet：通信对端的主机是具有公网IP的主机。

dest_ip_nat_count：通信对端是NAT后的内网主机数。

在接收到通信对端的数据包后，使用前面介绍的识别对端类型算法来确定其对端类型，只要通信对端是NAT后的内网主机数大于判断阈值 n_4，则认为该IP产生的数据流是P2P流。

定义 3.4 通信对端类型特征判断函数

$$\text{dest\_ip\_nat\_count} > n_4 \tag{3-2}$$

3. P2P-CNTIM 特征判断函数定义

通信对端类型特征和多主机特征对于P2P-CNTIM来说是或的关系，即只要任何一个特征条件得到满足就可以判断出该数据流是P2P流。

定义 3.5 P2P-CNTIM特征判断函数

$$\text{dest\_ip\_nat\_count} > n_4 \;\|\; \begin{cases} \text{diff\_dest\_ip\_count} > n_1 \\ \text{diff\_dest\_port\_count} > n_2 \\ \text{diff\_src\_port\_count} > n_3 \\ \text{diff\_dest\_port\_count} \approx \text{diff\_dest\_ip\_count} \end{cases} \tag{3-3}$$

3.2.2 P2P-CNTIM 调度机制

对于一段时间内收集的数据包，经过统计后可以根据上面公式的条件是否满足来进行P2P流的识别，该公式决定了统计的变量以及统计过程的目标，也是P2P-CNTIM的核心判断函数，在确定了P2P流量识别的判断函数后，还需要进一步确定该模型的调度机制[15]。

定义 3.6 P2P-CNTIM调度控制相关变量

packet_valid_time：数据包的有效时间。

recv_packet_time：数据包接收到的时间。

limit_bandwidth：用户网络带宽限额。

control_time：对目标主机的P2P流量控制时长。

p2p_interval_time：目标主机连续被识别出P2P流的时间间隔。

constant_value：调整流量控制时长中使用的常量。

对于收集到的某个内网主机的数据包来说，一段时间内的数据包只表明了这段时间该主机的网络行为，所以每个数据包从收集到的开始时间计算，经过一段时间后，再使用该数据包进行统计就没有意义[16]。因此，系统需要定义数据包的有效时间 packet_valid_time，如果当前时间减去收到数据包时间 recv_packet_time 大于 packet_valid_time，对于统计算法来说，该数据包就是无效的，如果把从某个内网主机的通信的数据包按照时间顺序进入队列，那么 packet_valid_time 决定了这个队列的长度，packet_valid_time 如果太长，则队列太大，且很多数据包只能代表该网络主机过去的行为，得到的统计结果就有可能滞后，如果 packet_valid_time 太短，则收到的数据包的个数太少，统计的样本太少，同样影响统计的结果，所以需要选择合适的 packet_valid_time。

进行 P2P 流量识别的根本目的是为了限制 P2P 应用对带宽的占用，在网络系统规划时，一般都预先核算了用户对带宽的要求，即用户平均带宽使用量 limit_bandwidth，可以使用该值来作为 P2P-CNTIM 的调度驱动，如果用户当前网络应用占用的带宽小于该用户的 limit_bandwidth，则没有必要启动 P2P 流量识别处理，只有当用户当前网络应用占用带宽大于 limit_bandwidth 时，系统的 P2P 流量识别处理才被激发，以检查该主机是否使用了 P2P 业务，如果统计表明使用了 P2P 业务，则对该用户的通信连接进行带宽限制。

内网主机被识别出使用了 P2P 业务后，识别系统需要对该主机的网络行为进行控制，从识别出 P2P 流的时间开始，在一个确定的时间范围 control_time 之内，该目标主机将被 P2P 识别系统进行协议阻断，以对该主机的带宽使用进行限制，对于每个内网 IP 主机的 control_time 是一个动态的变化过程，会根据用户的网络使用情况进行调整，调整的算法根据取消对内网主机的带宽使用限制后，到再次发现该主机产生 P2P 流之间的时间间隔 p2p_interval_time 来决定如何调整 control_time。

定义 3.7　P2P-CNTIM 流量控制时长调整计算

$$\begin{cases} \text{temp\_times} = \text{p2p\_interval\_time}/\text{constant\_value} \\ \text{control\_time} = \text{control\_time}/\text{temp\_times} \\ \text{control\_time} = \max(\text{default\_control\_time}, \text{control\_time}) \end{cases} \tag{3-4}$$

式(3-4)中的第一式计算出某个内网主机从解除限制到再次发现 P2P 流的时间间隔 p2p_interval_time 和系统定义的常数 constant_value 之间的倍数 temp_times，如果 temp_times＞1，则式(3-4)的第二式使得 control_time 变小，如果 temp_times＜1，则式(3-4)的第二式使得 control_time 变大。这就是说 p2p_interval_time 越大，则 control_time 越小；而 p2p_interval_time 越小，则 control_time 越大，式(3-4)中的第三式使得 control_time 不能小于系统定义默认最小值。

通过上面的算法，就可以动态地根据用户使用 P2P 流的情况进行区别限制，越频繁使用 P2P 应用的用户受到的控制时间就越多。

3.2.3　P2P-CNTIM 核心过程

在获得了 P2P-CNTIM 的特征判断函数以及调度机制之后，就可以方便地给出本章提出的 P2P 流量识别模型用伪代码表示的核心过程[2,17,30]。

在图 3-11 定义的数据结构中，其中 nat_test_status 列表存放系统关于通信对端主机类

型的测试状态，对于被观察主机通信的每一个对端 IP 地址，测试的过程和结果都存放在该列表中，由单独的通信对端类型检测模块维护该数据结构，该数据信息被所有的被检测主机共享，即在确定一个被检测主机的对端类型时，只需要查找该数据结构即可。如果在该列表中找不到目标对端主机对应的信息，则通知对端类型检测模块对目标对端主机类型进行检测。

```
/*数据结构定义*/
Define queue(i) is the packet queue of host(i);
Define p2pstream(i):
        if host(i) is producing p2p stream,then
        p2pstream(i)=true;else p2pstream(i)=false;
Define feature_multi_host(i):
        if host(i) satisfy formula 1 then
        feature_multi_host(i)=true else
        feature_multi_host(i)=false;
Define feature_nat_host(i):
        if(host(i) satisfy formula 2 then
        feature_nat_host(i)=true else
        feature_nat_host(i)=false;
Define control_time(i) :
        time to control host(i) if p2pstream(i)=true;
Define nat_test_status(i):
        0-is testing;1-nat device; 2-Internet host;3-test failed
limit_bandwidth(i):
        bandwidth limit of host(i)
```

图 3-11　数据结构定义

在图 3-12 的处理过程中，P2P 流检测系统为每个被检测主机建立一个数据包队列，在接收到数据包时，即进行入队列，然后把失效的数据包从队列中移除。

```
/*被检测主机数据包队列维护的处理过程*/
Func_add_packet_to_queue(host i, packet k)
{
   insert_time=current_time = time();
   pair temp=(k,insert_time); //组成二元组
   add temp to the head of queue(i);  //加入队列
   tail = queue(i).size;
  /*删除失效的数据包*/
  while(current_time-queue[tail].insert_time > packet_valid_time)
  {
       delete queue[tail];
       tail = queue[i].size;
  }
}
```

图 3-12　被检测主机数据包队列维护的处理过程

在图 3-13 的处理过程中，根据被检测主机的数据包队列计算是否满足式(3-1)，如果满足，则将多主机特性对应的变量置为 true。

在图 3-14 的处理中，计算被检测主机每个对端主机所属的类型，根据 nat_test_status 列表中的信息设置其为不同的类型，如果在 nat_test_status 列表中不存在，就向对端类型检测模块发送消息，触发其开始检测该目标 IP 的对端类型，由于检测需要一定的时间，因此处理认为该对端主机类型目前无法确定，直接进行下面的对其他对端主机类型的检测，如果获

```
/*统计被检测主机是否符合多主机特征的处理过程*/
Func_statistic_feature_multi_host(host i)
{
    get diff_dest_ip_count from queue(i);
    get diff_dest_port_count from queue(i);
    get  diff_src_port_count from queue(i);
    if satisfy formula 1
        feature_multi_host(i)=true
    else
        feature_multi_host(i)=false;
}
```

图 3-13　统计被检测主机是否符合多主机特征的处理过程

```
/*确定通信对端类型特征的处理过程*/
Func_statistic_feature_nat_host(host i)
{
    dest_ip_nat_count=0;
    for each different other host k of queue(i)
    {
        if(host k is in nat_test_status)
        {
            switch(nat_test_status(k))
            {
                case 0:continue;
                case 1:dest_ip_nat_count++;continue;
                case 2:continue;
                case 3:continue;
            }
        }
        else
        {
            add host k to nat_test_status;
            nat_test_status(k).status=0;
            notify nat test module to start nat test of host k;
            continue;
        }
    }
    if satisfy formula 2
        feature_nat_host(i)=true;
    else
        feature_nat_host(i)=false;
}
```

图 3-14　检测通信对端类型特征是否满足的处理过程

得的 dest_ip_nat_count 满足式(3-2),流检测系统就判定该主机产生了 P2P 流。

图 3-15 给出了 P2P-CNTIM 处理过程,该过程调用了处理子过程,整个 P2P 流检测模型是数据包驱动的,只要在检测设备上接收到数据包,就需要调用该函数完成 P2P 流的识别处理,该函数首先调用 Func_add_packet_to_queue 将收到的数据包入队列,如果该数据包对应的主机已经被标识为产生 P2P 流的主机,就直接返回,因为在以前的处理中已经通知过网络控制模块进行控制了;否则,由于队列中是保存的被检测主机一段时间的数据包,可根据该数据包中数据字节的总数来作为该主机当前占用的带宽,如果该带宽值超过了该主机的带宽限制,则启动 P2P 流量识别进行检测。函数首先调用 Func_statistic_feature_nat_host 来计算该主机是否满足通信对端类型特征,如果失败,则调用 Func_statistic_

feature_multi_host 来进行多主机特征的检测，只要这两个函数中的任一函数得到的结果满足式(3-1)或者式(3-2)，则说明被检测主机产生了 P2P 流量，将该主机对应的 p2pstream 变量置为 true 后，通知网络流量控制模块对该主机进行流量控制。

```
/*P2P流检测处理过程*/
Func_p2p_stream_check(host i, packet k)
{
    Func_add_packet_to_queue(host i, packet k);
    if(p2pstream(i) == true)//已经检测出，正在控制
    {
        return;
    }
    /*检查该主机占用的带宽是否超过限制*/
    total_packet_size = sum(each packet size of queue(i));
    if(total_packet_size > limit_bandwidth(i))
    {
        Func_statistic_feature_nat_host(host i);
        if(feature_nat_host(i)== false)
        {
            Func_statistic_feature_multi_host(host i);
        }
        if(feature_nat_host(i)== true || feature_multi_host(i)==true)
        {
            p2pstream(i) = true;
            /*属于P2P流通知控制模块对主机I进行流量控制*/
            notify p2p control module to control host  i
        }
    }
}
```

图 3-15　P2P 流检测处理过程

此外，P2P 流检测系统中还有通信对端类型检测模块的处理算法，该算法在 3.2.3 节中已经给出，系统在流量控制模块中将根据式(3-4)来调整 control_time 的时间，由于控制部分不是 P2P-CNTIM 的重点，所以不再详述。

3.3　P2P-CNTIM 系统的设计

3.3.1　P2P-CNTIM 系统的功能

满足 ISP 和企业网络管理需要的 P2P 识别、控制管理原型系统应该具备如下的基本功能，它们也是本系统的核心功能[18]。

1. P2P 流量识别

基于 Payload 特征来进行 P2P 应用识别的技术具有简单、可靠的优点，是目前 P2P 应用识别广泛使用的方法，然而对于新出现的 P2P 应用，人们分析出其协议特征往往需要一段时间，且该技术对于加密的 P2P 协议无法适用，所以采用基于 Payload 特征来进行 P2P 应用识别检测出来的 P2P 流仅是实际 P2P 流的子集。本原型系统要求能够对未知类型的 P2P 应用产生的 P2P 流进行识别，即首先通过 P2P 流量识别方法将数据流划分为 P2P 流和

非 P2P 流，然而在对 P2P 流进行应用细分[19]。

2. P2P 应用识别

网络上的 P2P 流是由各种具体的 P2P 应用产生的，在获知 P2P 流的情况下，需要进一步获取其具体的 P2P 应用类型，以便根据不同的应用采用不同的控制管理手段，同时系统应该具有很好的扩展性，能够随着各种 P2P 应用的发展检测出更多的 P2P 应用类型。

3. P2P 流控制管理

在识别出 P2P 流和 P2P 应用后，为了减少 P2P 流对带宽的过度占用，需要采用一定的技术来对 P2P 流进行控制。例如，利用速率限制将网络出口的 P2P 流量限制在一定的带宽之内；利用时段策略，在不同时段对 P2P 流量有不同的限制，以减少在其他重要任务和带宽敏感应用运行时所引起的网络阻塞。

3.3.2　P2P-CNTIM 系统结构

P2P 识别、控制管理系统需要首先实现对 P2P 流进行识别的功能，在从数据流中识别出 P2P 流之后，再进一步识别出具体的 P2P 应用，P2P 流量识别是整个系统的基础，也是系统的核心组成部分，只有正确、有效地识别出 P2P 流，才能保证 P2P 应用识别和 P2P 控制管理的正确性。在前面已经论述了 P2P-CNTIM 的有效性和正确性，并给出了模型的判断函数、调度机制和核心过程，所以 P2P 识别、控制管理系统的 P2P 流量识别部分将使用 P2P-CNTIM 来实现。

通过对数据包应用层协议的检测发现 P2P 应用的方法具有简单、可靠的特点，从 P2P 识别技术相关产品部分中可以看出，目前绝大部分的产品使用 DPI 技术来对 P2P 应用进行具体识别，所以 DPI 技术是 P2P 应用识别的主流技术。因此，本系统采用基于特征字检测技术来实现 P2P 应用识别功能。

在识别出具体的 P2P 应用之后，根据数据包中的 IP 地址，就可以对该 IP 地址产生的数据流进行控制。控制系统可以和防火墙结合，在得到这类软件的节点 IP 地址后在防火墙上进行封杀，也可以使用协议阻断来进行流量控制，协议阻断主要利用欺骗技术，该技术对局域网内部主机有较快速的反应和控制效果，作为 P2P 识别、控制管理的原型系统本文只使用协议阻断来进行 P2P 流的控制管理。

在确定了实现系统功能的关键技术后，就可以确定原型系统的结构。

在图 3-16 中，P2P 识别、控制管理系统结构分为 4 个模块，分别是数据包提取分析模块、P2P 流量识别模块、P2P 应用识别模块、P2P 控制管理模块。数据包提取分析模块完成网络数据包的捕获和分类，经过协议分析后分别形成 TCP 数据包队列、UDP 数据包队列、ICMP 数据包队列。P2P 流量识别模块包含基于多主机特征的统计分析子模块以及基于通信对端类型特征的检测子模块。Payload 分析模块根据特征字来对 P2P 流进行具体的应用分类。P2P 流量管理控制模块完成对 P2P 流进行协议阻断以控制 P2P 流。

P2P 识别、控制管理系统对一个的数据包处理是一个串行的过程，整个处理流程如下。

(1) 在网络上接收到一个数据包时，调用数据包提取模块将其进入到队列，每个内网主机对应一个独立的 TCP 和 UDP 数据包队列，而 ICMP 数据包只设置一个全局队列。

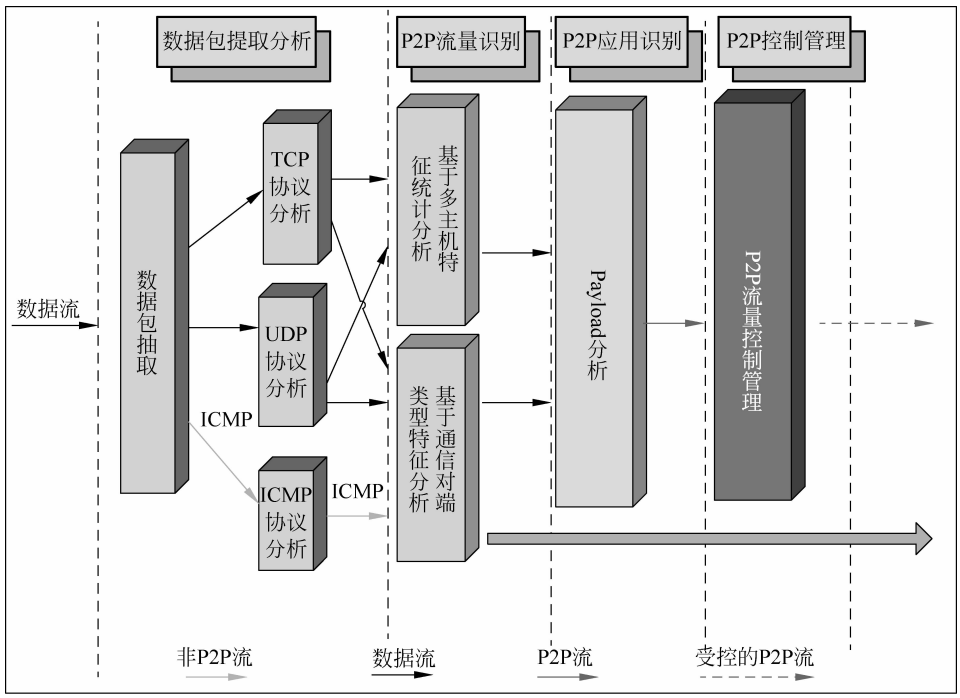


图 3-16　P2P 识别和控制管理系统结构

(2) 如果数据包为 ICMP 或者 TCP 的 RST 报文,则通知通信对端类型检测模块进行处理,通信对端类型检测模块是一个独立的模块,在本系统中使用单独的工作者线程进行处理。

(3) 如果是 TCP 或者 UDP 数据包,则调用 P2P 流量识别模块进行处理,P2P 流量识别模块如果检测到其没有被标识为 P2P 流但是超过了规定的带宽使用,则进行 P2P 流量识别,并在全局共享的信息 DataStreamClass 上对 iClass 标识其识别的结果。

(4) 调用 P2P 应用识别模块,如果 iClass 已经对应了具体的 P2P 应用,则处理结束,如果到该数据包对应的主机被标识为 P2P 流,即 iClass 被标识为 P2P 流,则进行 Payload 匹配处理,并根据匹配的结果对 DataStreamClass 上对 iClass 进行重新标识。

(5) 调用 P2P 控制管理模块,如果全局共享的信息 DataStreamClass 中的 iClass 为 P2P 流,且控制开始时间还小于 control_time 时间,则对数据流进行控制管理。

根据系统目标中的分析,基于 P2P-CNTIM 的 P2P 识别、管理控制系统是为了满足 ISP 和企业网络管理需要的网络管理系统,对于企业来说,P2P 识别、管理控制系统一般部署在企业的中心交换机附近,对于 ISP 来说,系统应该部署在各个家庭或者企业的汇集路由器或者交换机附近,在交换机上通过端口镜像将被监控端口的网络数据复制到 P2P 识别、控制管理主机上。系统运行部署如图 3-17 所示。

由于实验室条件限制,无端口镜像功能的交换机,因此原型系统通过将运行 P2P 识别、控制管理软件的主机的以太网卡设置为杂乱(Promiscuous)模式来监控实验室局域网内的所有主机的通信行为,这和端口镜像完成的功能是相同的,都能够获取到整个局域网内所有主机的网络流量,不影响原型系统的功能以及系统测试。

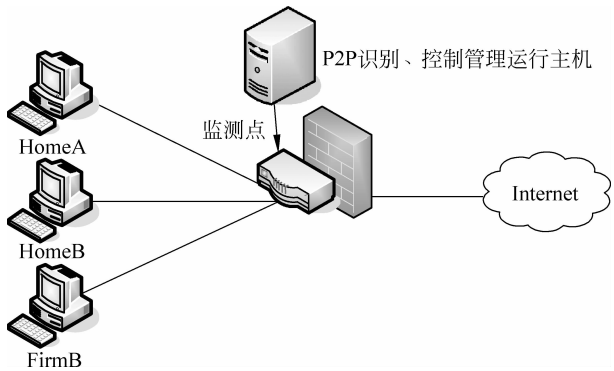


图 3-17　P2P 识别、管理控制系统部署位置图

3.4　P2P-CNTIM 系统的实现

3.4.1　数据包提取分析模块

数据包提取分析模块的功能就是从网卡上捕获局域网的主机通信数据，并分别按照TCP、UDP、ICMP 报文形成队列，每个报文都有一定的失效时间，每次在接收到数据包入队列时，同时删除队尾已经失效的数据包。在 Windows 平台下，系统采用成熟的 WinPcap 开发包来实现整个局域网内的数据包的捕获。[20]

1. WinPcap 技术简介

WinPcap 是由伯克利分组捕获库派生而来的分组捕获库，它是在 Windows 操作平台上来实现对底层包的捕获并且可以进一步的处理(根据用户的规则进行过滤或进行网络协议分析)。WinPcap 主要由内核层的包过滤(Kernel-level Packet Filter)、底层的动态连接库(Low-level Dynamic Link Library)和具有系统独立性的高层函数库(High-level and System-independent Library)三部分构成。WinPcap 的层次结构图，如图 3-18 所示。

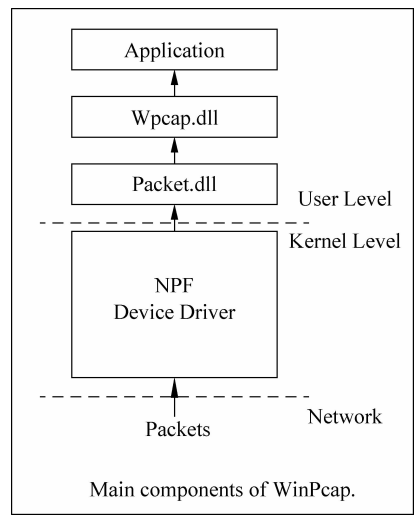


图 3-18　WinPcap 层次结构图

NPF(Netgroup Packet Filter)是一个虚拟设备驱动程序文件。它的功能是过滤数据包，并把这些数据包原封不动地传给用户态模块，这个过程中包括了一些操作系统特有的代码。

Packet.dll 为 Win32 平台提供了一个公共的接口。不同版本的 Windows 系统都有自己的内核模块和用户层模块。Packet.dll 用于解决这些不同。调用 Packet.dll 的程序可以运行在不同版本的 Windows 平台上，而无须重新编译。

Wpcap.dll 是不依赖于操作系统的。它提供了

更加高层、抽象的函数。Packet.dll 直接映射了内核的调用。Wpcap.dll 提供了更加友好、功能更加强大的函数调用。WinPcap 的优势在于提供了一套标准的抓包接口，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，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支持，支持内核态的统计模式，提供了发送数据包的能力。

2. 数据包提取分析模块处理流程

数据包提取分析模块处理流程如图 3-19 所示。

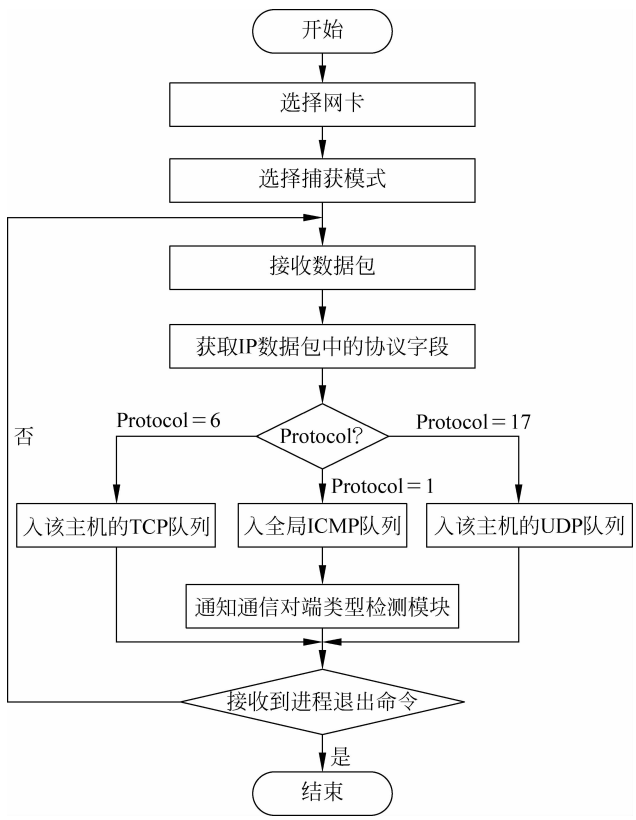


图 3-19　数据包提取分析模块处理流程图

数据包提取分析模块首先选择网卡，然后设置捕获的模式，在接收到数据包后，根据 IP 数据包中的协议字段来分别提取 ICMP、UDP、TCP 报文，在 TCP/IP 协议中，协议字段 1 表示 ICMP 协议，2 表示 IGM 协议，6 表示 TCP 协议，17 表示 UDP 协议，在接收到 ICMP 或 TCP 的 RST 报文后，该模块通知通信对端类型检测模块对该报文进行处理，对于 TCP 数据包如果是建立连接过程，但是没有产生数据流的，则不入 TCP 报文队列，对于传输数据的 TCP 和 UDP 数据包，则进入相应的主机报文队列，由于原型系统是数据包驱动的，因此在入主机数据包的队列后会首先删除失效的数据包，然后检查该主机是否已经被判定为产生 P2P 流的主机，如果不是，则检查其带宽占用，如果超过了占用的带宽，则触发 P2P 流检测的后续处理。

3. 关键数据结构定义

1）网卡信息结构体

```
typedef struct _ASTAT_
{
    ADAPTER_STATUS adapt;              //某个网卡的相关信息
    NAME_BUFFER NameBuff[30];          //网卡的名字信息
}ASTAT, * PSTAT;
```

2）IP 地址结构体

```
typedef struct ip_address
{
    u_char byte1;                      //IP 地址第一字节
    u_char byte2;                      //IP 地址第二字节
    u_char byte3;                      //IP 地址第三字节
    u_char byte4;                      //IP 地址第四字节
}ip_address;
```

3）IP 报头结构体

```
typedef struct ip_header
{
    u_char ver_ihl;                    //4bit 的版本信息 + 4bits 的头长
    u_char tos;                        //TOS 类型
    u_short tlen;                      //总长度
    u_short identification;            //Identification
    u_short flags_fo;                  //Flags(3bits) + Fragment offset (13bits)
    u_char ttl;                        //生存期
    u_char proto;                      //后面的协议信息
    u_short crc;                       //校验和
    ip_address saddr;                  //源 IP
    ip_address daddr;                  //目的 IP
    u_int op_pad;                      //Option + Padding
}ip_header;
```

4）TCP 包头结构体

```
typedef struct tcp_header
{
    u_short sport;                     //源端口
    u_short dport;                     //目的端口
    u_int seq;                         //序号
    u_int ack_seq;                     //确认号
    u_int16_t doff: 4;                 //报头长度
    u_int16_t res1: 4;                 //保留字节
    u_int16_t res2: 2;                 //保留字节
    u_int16_t urg: 1;                  //指示紧急指针
    u_int16_t ack: 1;                  //指示确认号
    u_int16_t psh: 1;                  //指示数据处理方式
    u_int16_t rst: 1;                  //指示复位连接
```

```
    u_int16_t syn: 1;                    //指示请求连接
    u_int16_t fin: 1;                    //指示数据已发完
    u_short window;                      //窗口
    u_short check;                       //校验和
    u_short urg_ptr;                     //紧急指针
}tcp_header;
```

5）UDP 头(udp_header)结构体

```
typedef struct udp_header
{
    u_short sport;                       //源端口
    u_short dport;                       //目的端口
    u_short len;                         //数据包长度
    u_short crc;                         //crc 校验值
}udp_header;
```

4. 关键类设计

1）网卡设置数据包捕获类

```
/* 该类完成主机网卡的选择,获取数据包并根据数据包的类型插入到对应的队列 */
class InterfaceCard
{
public:
    InterfaceCard();
    Virtual ~ InterfaceCard();
public:
    bool        SetFilter(char * packet_filter);          //设置过滤字符串
    void        SetNetmask();                             //设置子网掩码
    bool        CheckLink();                              //检查捕获设置
    bool        SelectCard(int inum);                     //选择捕获网卡
    u_int       netmask;                                  //子网掩码
    pcap_if_t    * allDevices;                            //网卡设备数组
    int         cardNum;                                  //网卡的数目
    pcap_if_t    * selectDevices;                         //选择的网卡设备
    pcap_t       * adhandle;                              //打开设备的句柄
    char        errbuf[PCAP_ERRBUF_SIZE];                 //存放错误信息
struct          bpf_program fcode;                        //过滤代码
static UINT     DoCaputer(LPVOID paramator);              //捕获数据包
int             add_packet_to_queue(char * packet);       //将数据包根据类别入队列
};
```

2）主机网络信息处理类

```
/* 该类用于获取主机上的网络信息 */
class HostNetInfo
{
public:
    HostNetInfo();
    virtual ~HostNetInfo();
    void GetMac(int AdapaterIndex);          //获取 MAC 地址
```

```
    void GetAdaperInfo();                    //得到当前正在使用的 IP 信息
    void GetAllMacAddr();                    //获得所有 MAC 地址
private:
    string strIP;                            //IP 地址
    string strNetMask;                       //子网掩码
    string strNetNum;                        //网络号
    };
```

3.4.2　P2P 流量识别模块

1. 模块设计

在前面已经详细给出了流量识别模型 P2P-CNTIM 的判断函数、调度机制以及核心过程。根据 P2P-CNTIM 中的描述，P2P 识别、控制管理系统的 P2P 流量识别模块分为多主机特征统计分析子模块、通信对端类型特征分析子模块、通信对端类型检测子模块以及 P2P 流量识别子模块共 4 个模块。

在图 3-20 中，箭头表示子模块之间的调用关系，P2P 流量识别子模块调用多主机特征统计分析子模块和通信对端类型特征分析子模块来获得对数据流的划分，通信对端类型特征分析子模块通过查询通信对端类型检测子模块的检测结果来决定该数据流是否是 P2P 流。

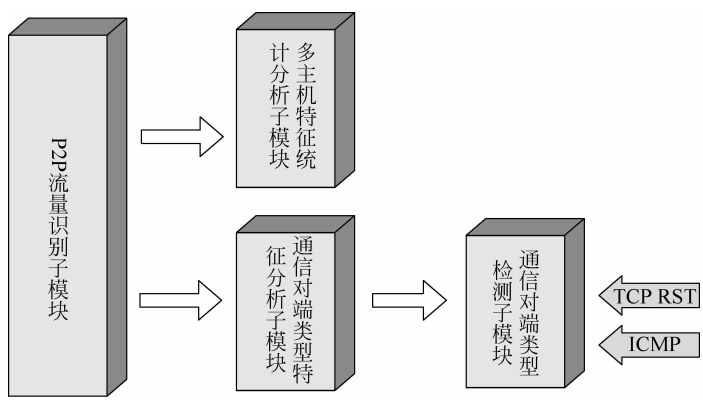


图 3-20　P2P 流量识别模块分解图

1) 通信对端类型检测子模块

该子模块是一个独立的子模块，维护整个内网的通信对端类型数据结构 nat_test_status 并进行更新，当通信对端类型特征分析子模块在其维护的数据结构 nat_test_status 列表中找不到对应的信息时，则向该子模块发送消息让该子模块启动对目标 IP 地址的通信对端类型检测，传入的参数是目标 IP 地址及其收到数据包的 TTL 值，该模块使用获取通信对端类型关键技术中提供的算法来检测通信对端类型，并根据检测的结果修改数据结构 nat_test_status 中的相关项。

输入：通信对端 IP，从该 IP 收到的数据包的 TTL 值。

输出：通信对端类型数据结构 nat_test_status。

2）通信对端类型特征分析子模块

该子模块对于被检测的内网主机中的每一个通信端口号大于10000的对端主机，根据其IP从通信对端类型检测子模块维护的数据结构nat_test_status中查找该主机所属的类型情况，如果在nat_test_status列表中找不到对应IP的信息时，则向该子模块发送消息让该子模块启动对目标IP地址的通信对端类型检测，传入的参数是目标IP地址及其收到数据包的TTL值。如果在nat_test_status列表中存在该IP对应的信息，由于nat_test_status(IP)的值可能为0-is testing；1-nat device；2-Internet host；3-test failed。只要被检测主机的对端nat_test_status(IP)的值为1-nat device的对端主机个数大于阈值参数，即满足式(3-2)，则认为被检测主机产生了P2P流，测试结束。该子模块的处理和图3-14检测通信对端类型特征是否满足的处理过程的伪代码类似。

输入：被检测主机的数据包队列，nat_test_status列表。

输出：被检测主机通信对端类型特征是否满足。

3）多主机特征统计分析子模块

该子模块根据被检测主机的数据包队列中的统计信息是否满足流量识别模型P2P-CNTIM中定义的式(3-1)来对数据流进行检测，处理过程中要分别统计被检测主机对应的UDP和TCP数据包队列中的不同目的IP、不同目的端口、不同源端口，如果满足式(3-1)，则说明被检测主机对于多主机的特征是满足的，也就是被检测主机产生了P2P流。该子模块的处理和图3-13统计被检测主机是否符合多主机特征的处理过程的伪代码类似。

输入：被检测主机对应的UDP和TCP数据包队列。

输出：被检测主机多主机特征是否满足。

4）P2P流量识别子模块

P2P流量识别子模块是识别模块中的调度模块，由它去触发其他模块的执行，P2P流量识别子模块本身在数据包提取分析模块每收到一个TCP或者UDP数据包时被触发调用，如果该数据包对应的主机已经被标识为产生P2P流的主机，就直接返回。否则，由于队列中保存的是被检测主机一段时间的数据包，可以根据该数据包中数据字节的总数来作为该主机当前占用的带宽，如果该带宽值超过了该主机的带宽限制，则启动P2P流检测。首先调用通信对端类型特征分析子模块来计算该主机是否满足通信对端类型特征，如果不满足，则调用多主机特征统计分析子模块来进行多主机特征的检测，只要这两个特征任何一个满足P2P流的特征，则说明被检测主机产生了P2P流量。其处理流程图如图3-21所示。

输入：被检测主机Host对应的数据包队列，nat_test_status列表。

输出：被检测主机是否产生了P2P流。

2. 关键数据结构定义

1）UDP数据包信息结构

```
struct UdpPktInfo
{
    string srcIP;                          //源地址
    string desIp;                          //目的地址
    UINT  srcPort;                         //源端口
    UINT  desPort;                         //目的端口
```

```
    UINT   pkt_len;                          //数据包长度
    UINT   pkt_appLen;                       //数据包负载长度
    basic_string<u_char> appData;            //负载数据
    time_t InsertTime;                       //数据包插入时间
};
```

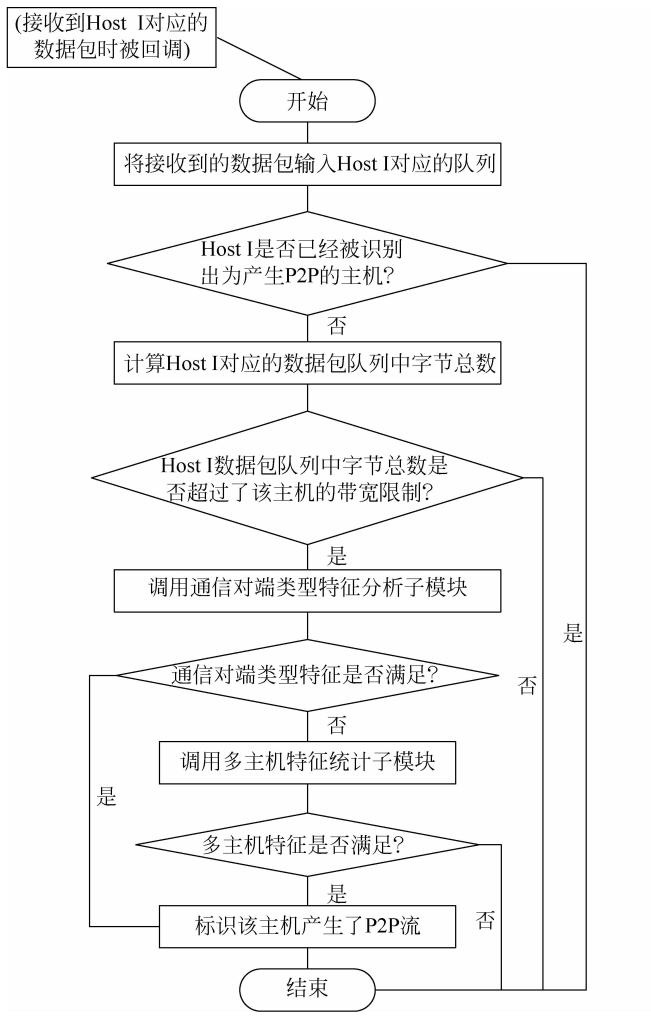


图 3-21　P2P 流量识别模块处理流程图

2）TCP 数据包信息结构

```
struct TcpPktInfo
{
    string srcIP;                            //源地址
    string desIp;                            //目的地址
    UINT   srcPort;                          //源端口
    UINT   desPort;                          //目的端口
    UINT   pkt_len;                          //数据包长度
    UINT   pkt_appLen;                       //数据包负载长度
    basic_string<u_char> appData;            //负载数据
```

```
    bool syn;                          //SYN 标志
    bool ack;                          //ACK 标志
    time_t InsertTime;                 //数据包插入时间
};
```

3）ICMP 数据包信息结构

```
struct ICMPPktInfo
{
    string  strSrcIP;                  //源地址
    string  strDestIP;                 //目的地址
    char    cType;                     //ICMP 类型
    char    cCode;                     //ICMP 代码
    short   sCheck;                    //ICMP 校验值
};
```

4）多主机统计特征信息结构

```
struct MultiHostFeatureInfo
{
    string localIP;                    //本地 IP,作为唯一索引
    UINT  diff_SrcPort;                //源端口数
    UINT  diff_DesIP;                  //目的 IP 数
    UINT  diff_DesPort;                //目的端口数
    UINT  Num_SYN;                     //SYN 标志有效的 < srcIP,desIP >
    UINT  Num_ACK;                     //ACK 标志有效的 < srcIP,desIP >
    list < UINT > diffSrcPorttemp;     //IP 源端口列表
    list < UINT > diffDesPorttemp;     //IP 目的端口列表
    list < string > diffdesIPtemp;     //IP 目的 IP 列表
};
```

5）通信对端类型结构

```
struct nat_test_status
{
    string  strTargetIP;               //通信对端 IP
    int        iStatus;        //0 - testing;1 - nat device; 2 - Internet host;3 - test failed
    list < int > listTTL;              //和内网主机传输的数据包不同 TTL 列表
    time_t     TestStartTime;          //检测开始时间
    time_t     TestFinishTime;         //检测结束时间
};
```

6）数据流类别信息结构体

```
struct DataStreamClass
{
    string  strIP;                     //被检测主机的 IP 地址,key
    int     iClass;                    //类别 0 - 非 P2P,1 - P2P,2 - BT,3 - PPLive…
    int     iControl_Time;             //控制时长
    bool    bControl_Time_Adjusted;    //控制时长是否已经被调整
    int     iBandwidth;                //分配带宽
```

```
    bool    Feature_Multi_Host;          //多主机特征
    bool    Feature_NAT_Host;            //对端类型特征
    time_t  StartControlTime;            //开始控制的时间
    time_t  EndControlTime;              //结束控制的时间
};
```

3. 关键类设计

1）对端类型检测类

```
/*该类用于实现通信对端类型的检测*/
class TestNatHost
{
public:
    TestNatHost();
    Virtual ~ TestNatHost();
public:
list<nat_test_status> listHostType;          //维护的全局对端类型列表
       int SendPingPacket(char* cpTargetIP);  //发送 ping 报文
       int SendTCPPacket(char* cpTargetIP);   //发送高端口 TCP 连接报文
       int SendUDPPacket(char* cpTargetIP);   //发送高端口 UDP 数据报文
       int DealICMPPacket(char* cpPacket, int iPacketLen);      //处理到该主机的 ICMP 报文
       int DealTCPRstPacket(char* cpPacket, int iPacketLen);  //处理到该主机的 TCPRST 报文
       int TestHostType(char* cpTargetIP);    //测试通信对端类型
   …
};
```

2）多主机特征检测类

```
/*该类用于实现对多主机特征的检测,以判断被检测主机是否产生了 P2P 流*/
class MultiHostFeatureTest
{
public:
    MultiHostFeatureTest();                  //构造函数
    Virtual ~ MultiHostFeatureTest();        //析构函数
private:
       int TCPStatistics(char* cpTargetIP);  //对目标主机统计 TCP 多主机特征
       int UDPStatistics(char* cpTargetIP);  //对目标主机统计 UDP 多主机特征
public:
   int TestMultiHostFeature(char* cpTargetIP); //对目标主机多主机特征进行检测
   …
};
```

3.4.3　P2P 应用识别模块

1. 模块设计

数据流类别信息结构体 DataStreamClass 中定义了被检测主机对应的数据流类别

iClass，在接收到 UDP 或者 TCP 数据包后，系统将调用 P2P 流量识别模块来判断该数据流是否是 P2P 流，如果其满足多主机特征或者通信对端的类型为 NAT 后的内网主机特征，则在 DataStreamClass 中将其 iClass 字段设置为 1，即该主机产生的数据流为 P2P 流。然后识别系统将调用 P2P 应用识别模块对该主机产生的数据流进行处理，如果 iClass 已经对应了具体的 P2P 应用，则应用识别模块直接返回，如果该数据包对应的主机被标识为 P2P 流，即 iClass 被标识为 P2P 流，但是还没有识别出对应的 P2P 流应用，则进行 Payload 特征匹配处理，并根据匹配的结果对 DataStreamClass 中 iClass 字段进行重新标识。其处理的流程如图 3-22 所示。

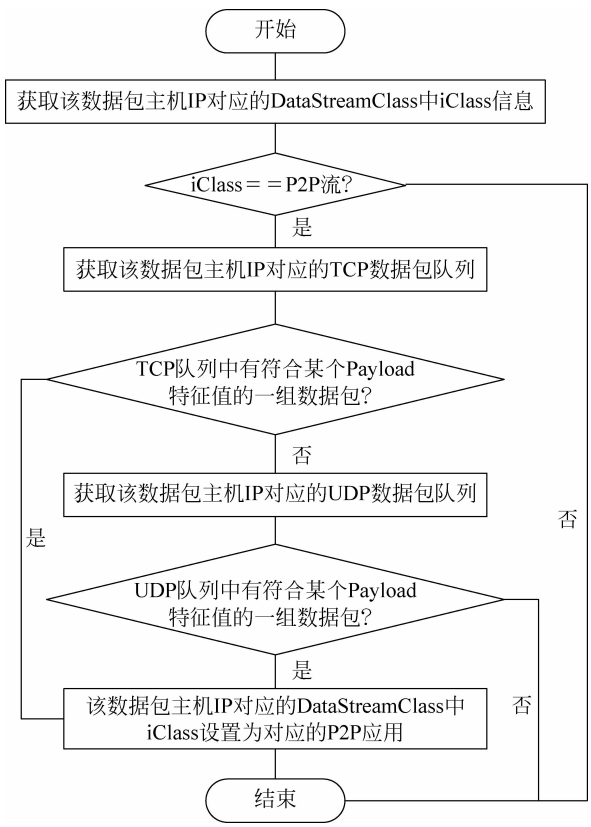


图 3-22　P2P 应用识别处理流程图

该模块在分别对 TCP 队列和 UDP 队列中 Palyload 特征值进行匹配识别，在系统实现时，为了加快识别的速度，提高识别效率，采用两个并发线程同时分别进行 TCP 和 UDP 的 Palyload 特征识别，并根据匹配的结果对 DataStreamClass 上的 iClass 进行标识。

各种具有不同 Payload 特征的 P2P 应用将会不断出现，考虑到扩展性，系统对每个应用的 Payload 特征统一在配置文件中配置，应用识别模块读取该配置文件形成 Payload 检测的规则，当检测出新的 P2P 应用的 Payload 特征时，只要把该新的 Payload 特征再增加到配置文件中即可。作为实际的应用系统，特征匹配是提高系统效率的核心所在，但是本系统作为原型系统，将不实现比较复杂的匹配模块，表 3-3 中列出了本原型系统使用的部分 Payload 特征。

表 3-3　部分 P2P 应用的 Payload 特征

序号	P2P 应用	特　征　值
1	BT	TCP 载荷第一个字节的值是“0X13”;接下来字节的值匹配于字符串“BitTorrentprotocol”
2	eDonkey,eMule	头两个字节为“E3”或“C5”或“D4”;接下来 4 个字节是 packet length,其值是 TCP 数据段的长度减去 5 个字节
3	Gnutell	关键字“67 6e 75 74 65 6C 6C 61”或者“47 6e 75 74 65 6c 6c 61”
4	DireetConnect	TCP 包头后载荷的前 4 个字符为“$Sen”、“$Get”或“$Fit”，最后一个为“ ”
5	FastTrack	关键字“58 2D 4B 61 7A 61 61”
6	PPLive	TCP：0x e9 03 (+0/+4) UDP：0x e9 03 * * 98 ab(+0)
7	PPStream	UDP：0x 00/01/02 00 00 03(+1) 0x 00/01/04 43 00 00(+1) TCP：PSProtocol(+0)‖0x 11 00 00 00 03(+0)
8	QQLive	UDP：0xfe * 00 00 *(+0)‖0xfe * 04 04 *(+0)
	…	…

2. 关键数据结构定义

1) 特征中包含的单元信息结构

```
struct FeatureUnit
{
    int iPosition;                          //报文中的位置
    int iLength;                            //数据长度
    basic_string<u_char> FeatureData;       //匹配数据
};
```

2) 特征信息结构

```
struct Feature
{
    int iPacketType;                        //数据包类型
    int iStartPos;                          //开始检测的位置
    int iCheckType;                         //全面检测还是固定位置检测
    std::list<FeatureUnit> listPart;        //由多个特征单元构成的特征
};
```

3) P2P 应用 Payload 结构

```
struct AppFeature
{
    string strAppName;                      //应用名称
    string strAppVersion;                   //应用版本
    std::list<Feature> listFeatures;        //特征列表
    string FeaturesLogic;                   //特征间的逻辑关系,目前支持与/或
};
```

3. 关键类设计

Payload 特征检测类设计如下。

```
/* 该类用于实现已经识别出的 P2P 流进行 Payload 特征的检测,以进一步区分 P2P 应用 */
class PayloadMatch
{
public:
    PayloadMatch ();                                    //构造函数
    Virtual ~ PayloadMatch ();                          //析构函数
private:
    std::list < AppFeature > listAppFeature;            //读取的应用特征列表
    int DoTcpMatch(char * cpTargetIP);                  //TCP 报文特征匹配
    int DoUDPMatch(char * cpTargetIP);                  //UDP 报文特征匹配
public:
    int ReadFeatures(char * cpFilePath);                //读取 P2P 应用特征信息
    int FeatureMatch(char * cpTargetIP);                //对 P2P 应用进行特征匹配
      …
};
```

3.4.4 P2P 控制管理模块

1. P2P 控制管理技术介绍

TCP 或者 UDP 数据包经过 P2P 流量识别模块、P2P 应用识别模块处理后,系统调度程序将调用 P2P 控制管理模块对该数据包进行处理,如果该数据包对应的主机被系统判断为产生 P2P 流的主机,P2P 控制管理模块将对该主机进行网络流量控制,实现网络流量控制的主要技术有以下几种。

1) TCP 异常阻断

TCP 协议栈对网络异常的处理优先级较高,通常 TCP 连接发生错误时,TCP 连接会产生异常中断,即 TCP 连接双方的任何一方发送一个复位报文段(TCP 首部中的 RST 比特置“1”,RST 包)来中途释放一个连接,应用程序 API 异常关闭。网络控制管理模块根据通信连接特征,构造 RST 数据报,来模拟 TCP 连接异常释放,使对端不再接收原通信方的数据,从而实现通信连接的阻断,但这种方法不够隐蔽,容易被发现[21]。

2) TCP 正常阻断

TCP 连接数据交换结束,将终止连接,也称为有序释放。终止一个连接要 4 次握手,即正常中止连接有 4 个交互数据报。发送 FIN 通常是应用层进行关闭的结果。在发现一个需要控制的连接后,利用连接信息构造 FIN 包,并分别向连接的双方发送,使得该连接提前结束,完成连接阻断。该技术利用正常的 TCP 连接结束进行阻断连接,没有明显的阻断特征,对端不易发现,不易防御[22]。

3) TCP 填充窗口阻断

利用协议栈中的滑动窗口协议,模拟服务器向客户端发送无用数据、占用正常 TCP 连

接的序列号，使服务器发送的数据不被客户端接收，或者打乱服务器发送的数据内容，在多次发生序列号冲突的情况下，协议栈会发送异常终止结束连接，以达到阻断连接的目的。

4）ICMP 欺骗阻断

ICMP 欺骗阻断分为网络不可达阻断、主机不可达阻断、协议不可达阻断及端口不可达阻断几种方式。网络控制管理系统模拟通信节点向对方发送一个 ICMP 报文，告诉它主机、协议或者端口不可达，从而达到阻断的目的。

5）ARP 欺骗阻断

在以太网中，一个主机要和另一个主机进行直接通信，必须要知道目标主机的 MAC 地址。ARP 协议的基本功能就是通过目标设备的 IP 地址，查询目标设备的 MAC 地址，以保证通信的顺利进行，网络控制管理系统可以使用 ARP 欺骗技术实现对节点进行攻击和封锁。

6）和防火墙联动阻断

网络控制管理系统与防火墙等设备进行联动。与防火墙设备联动需要防火墙开放相应的配置接口，目前众多商用防火墙和自由防火墙软件这一技术都已经实现的比较完善，成熟。

2. P2P 控制管理模块处理流程

在实际的应用系统中，需要综合采用上面的技术来实现对 P2P 流的控制管理，同时需要针对不同的 P2P 应用采用不同的控制管理策略，作为原型系统，为了减少系统的复杂度，主要使用 TCP 阻塞技术来实现对被监控主机的网络带宽控制。同时对于所有的 P2P 应用使用相同的控制管理方法。

P2P 控制管理模块首先检查数据包对应 DataStreamClass 中的 iClass 的类别[23]，如果不是非 P2P 流，而且当前时间减去开始控制的时间 StartControlTime 小于控制时长 iControl_Time，则采用阻断技术对该数据包进行控制，也就是采用 TCP 正常阻断技术，根据通信两端最后一个数据包中的序号伪造 FIN 报文，并分别向连接的双方发送，使得这两个主机之间的连接提前结束，完成连接阻断。如果当前时间减去开始控制的时间 StartControlTime 大于控制时长 iControl _ Time，则说明应该停止控制，修改 DataStreamClass 中的 iClass 为非 P2P 流，bControl _ Time _ Adjusted 设置为 false，EndControlTime 设置为当前的系统时间，在下一次检测出 P2P 流时，需要根据该从上一次控制结束到再次被判断为 P2P 流之间的时间间隔来调整控制时长 iControl_Time，整个处理流程如图 3-23 所示。

3. 关键数据结构定义

该模块主要使用 P2P 流模块识别中已经定义的数据结构。

1）数据流类别信息结构体

```
struct DataStreamClass;
```

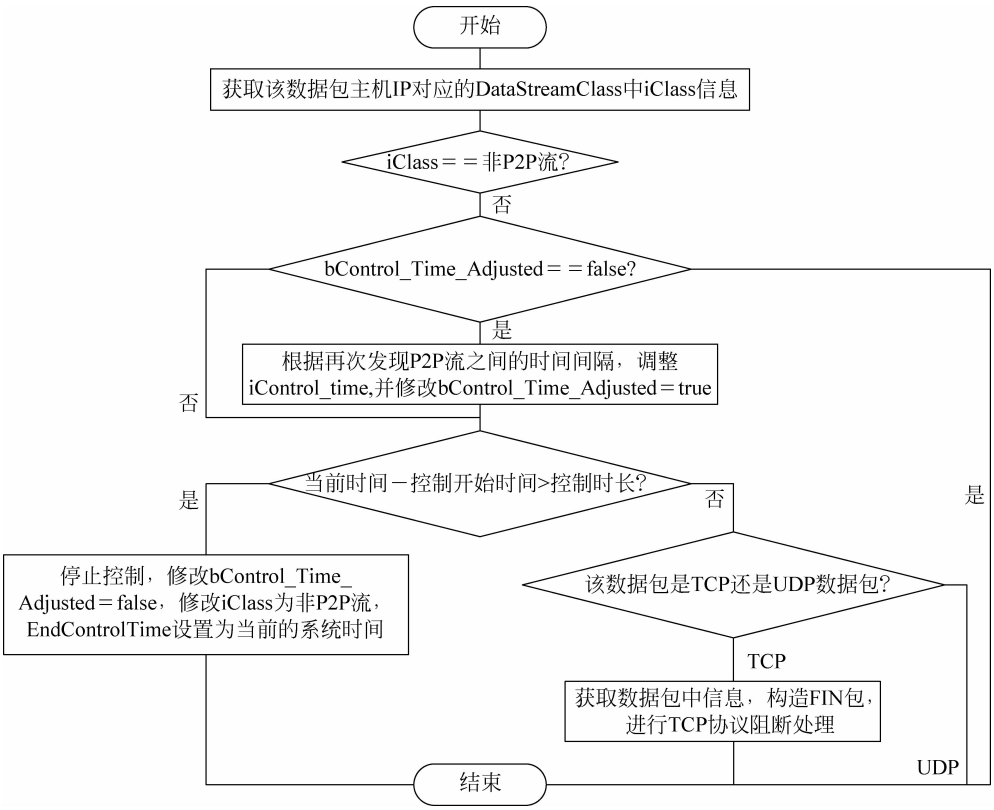


图 3-23　P2P 控制管理模块处理流程图

2）TCP 包头结构体

```
struct tcp_header;
```

4. 关键类设计

P2P 流量控制管理类设计如下。

```
/* 该类用于实现已经识别出的 P2P 流进行 TCP 协议阻断,以减少 P2P 应用占用带宽 */
class P2PControl
{
public:
    P2PControl();                                    //构造函数
    Virtual ~ P2PControl();                          //析构函数
public:
    int AdjustCtrolTime(char * cpIP);                //调整控制时间
    int FormatTCPFindPacket(tcp_header lastRecvHead);   //构造的 TCP 结束报文
    int SendRawPacket(char * cpSendPacket, int iSendLen);   //发送 TCP 报文
    int DoControl(char * cpIP);                      //流量控制入口
    int SetCotnrolEndInfo(char * cpIP);              //初始化控制信息
      …
};
```

3.5　P2P-CNTIM 系统测试

3.5.1　测试环境

为了测试基于 P2P-CNTIM 的 P2P 识别、控制管理原型系统的性能，在实验室环境下对原型系统进行了测试，由于目前的很多产品都是基于 Payload 特征技术来对 P2P 应用进行识别的，因此将软件的 P2P 识别模块抽取出来单独形成 P2P 应用识别软件，以进行比较。实验室内选择其中的 20 台主机组成了局域网，它们连接在实验室的交换机上，然后通过该交换机可以访问 Internet，检测系统部署在实验室的一台服务器上，在该主机上将网卡模式设置为混杂模式，以获得整个实验室所有主机的通信流量。实验主机的一部分用作 P2P 节点以产生足够大的 P2P 流量来检测这两种方法的鲁棒性和检测精度，另外一些主机运行 Web 服务以及 FTP 等传统网络应用，整个测试的实验网络环境如图 3-24 所示。

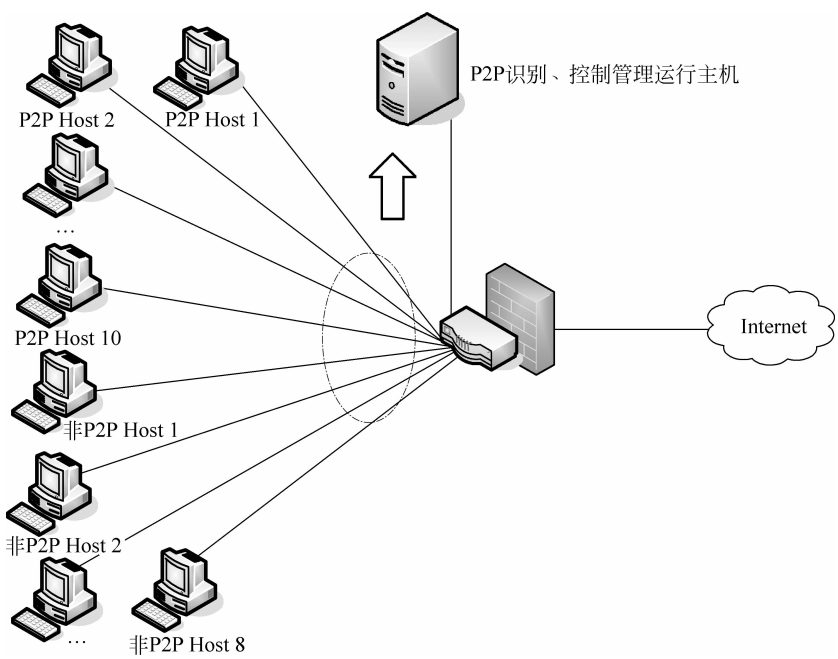


图 3-24　P2P-CNTIM 原型系统测试网络结构图

3.5.2　误判率测试分析

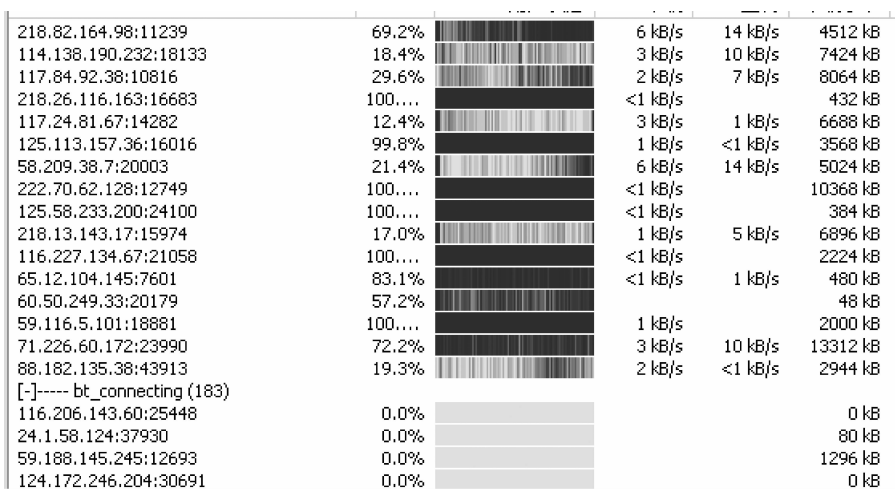
1. 基于多主机特征检测的误判率

在保证识别准确率的基础上，尽可能降低误判率是 P2P 流量识别模型的基本原则，根据 P2P 流量识别模型中的式(3-1)，当 diff_dest_ip_count$>n_1$，diff_dest_port_count$>n_2$，diff_src_port_count$>n_3$，diff_dest_port_count$\approx$diff_dest_ip_count 时多主机特征将把该数据流判别为 P2P 流，在开始实验前，需要讨论相关参数的确定。

首先需要确定数据包的有效时间 packet_valid_time，该值决定了数据包队列的大小，假设用户的带宽为 80kb/s，每个数据包的平均长度为 512 个字节，那么，在一个单位分钟内，统计的样本数据包的个数约为 80×1024/500×60＝9600 个数据包，那么在一个单位时间内，已经获得了足够的数据包来进行统计，所以将 packet_valid_time 设置为 1 个单位时间。

假设一个用户在极限的情况下，平均 1 秒内打开 1 个网页，那么在一分钟之内，最多能打开 60 个网页，也就是最多访问 60 个不同的 IP 地址，然而，进一步假设该用户以 50%的概率在同一个网页内打开链接，另一方面，用户打开的网页不使用标准的协议端口的只占很少的一部分，在这里假设使用不规则端口占 5%，那么普通的 Web 用户在一分钟之内访问的 IP 数为 60×50%＝30，不规则的端口数为 30×5%＝1.5 个，加上常规使用的端口如 80 端口、21 端口等，其不同的端口数一般不会超过 30%，即总的不同的端口数不会超过总的 IP 个数的 30%。

而在图 3-25 中，可以看到，P2P 应用往往使用多个不同的端口，对于 Bitcomet 来说，其端口特征有 90%的端口在 10000 以上，基本上一个 IP 就对应了一个不同的端口，根据对其他的 P2P 应用的实验，其不同的端口数占用的比例也是相当高的，这一部分是由于 NAT 设备的映射端口一般都大于 10000 造成的。



218.82.164.98:11239	69.2%		6 kB/s	14 kB/s	4512 kB
114.138.190.232:18133	18.4%		3 kB/s	10 kB/s	7424 kB
117.84.92.38:10816	29.6%		2 kB/s	7 kB/s	8064 kB
218.26.116.163:16683	100....		<1 kB/s		432 kB
117.24.81.67:14282	12.4%		3 kB/s	1 kB/s	6688 kB
125.113.157.36:16016	99.8%		1 kB/s	<1 kB/s	3568 kB
58.209.38.7:20003	21.4%		6 kB/s	14 kB/s	5024 kB
222.70.62.128:12749	100....		<1 kB/s		10368 kB
125.58.233.200:24100	100....		<1 kB/s		384 kB
218.13.143.17:15974	17.0%		1 kB/s	5 kB/s	6896 kB
116.227.134.67:21058	100....		<1 kB/s		2224 kB
65.12.104.145:7601	83.1%		<1 kB/s	1 kB/s	480 kB
60.50.249.33:20179	57.2%				48 kB
59.116.5.101:18881	100....		1 kB/s		2000 kB
71.226.60.172:23990	72.2%		3 kB/s	10 kB/s	13312 kB
88.182.135.38:43913	19.3%		2 kB/s	<1 kB/s	2944 kB
[-]----- bt_connecting (183)					
116.206.143.60:25448	0.0%				0 kB
24.1.58.124:37930	0.0%				80 kB
59.188.145.245:12693	0.0%				1296 kB
124.172.246.204:30691	0.0%				0 kB

图 3-25　典型的 P2P 应用 Bitcomet 运行时截图

根据上面的讨论，可以取 $n_2＝n_1×50\%$，$n_3＝n_1×30\%$，这样的公式显然一般不会对传统的网络应用造成误判，这样多主机特征公式将由通信对端不同的主机数量阈值 n_1 唯一确定，n_1 为单位分钟内基于多主机特征的判定阈值，通过对 HTTP 和 FTP 等非 P2P 流进行测试，以来确定合适的 n_1 的值。

在测试过程中，实验室的 8 台主机产生 Web、FTP 等传统的网络流量，在实验过程中，将每个用户的带宽使用限制设置为 25kb/s，设置较低的带宽限制是为了使得 P2P 流量识别模块能够基本保持在检测的状态，为了减少流量控制模块对实验误判率的影响，所以临时关闭了控制管理模块的功能，并且设定一个数据流被识别为 P2P 流后，其受控的时间为 1 个单位分钟，在到了受控时间后，将其再标识为普通的数据流，这样当一个数据流被标识为 P2P 流时，由于其数据包队列中包含了 1 个单位分钟的数据包，这部分的数据包可以认为是 P2P 流的数据包，同时由于受控的时间是 1 个单位分钟，因此后面的 1 个单位分钟内，该主

机的数据包也被认为是 P2P 流的数据包，将这两个单位分钟内的 P2P 流的数据包和这 8 台主机产生的总的数据包之间的比值作为误判率。当 n_1 取不同值时获得的对传统网络应用的误判率如图 3-26 所示。

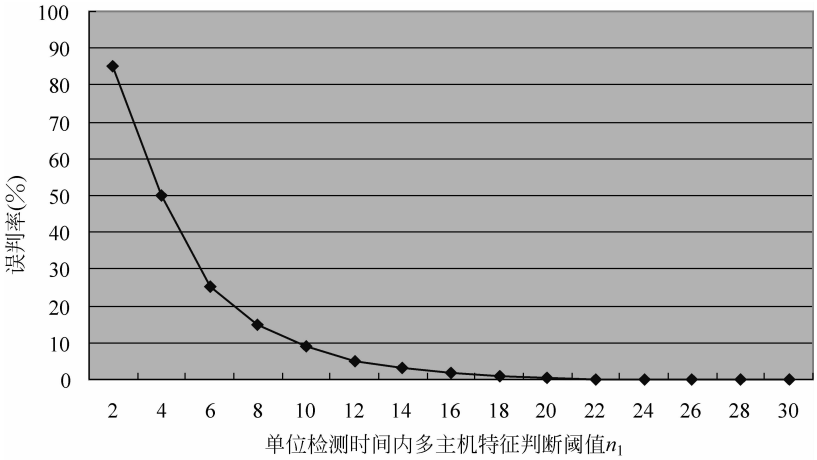


图 3-26　基于多主机特征的 P2P 检测误判率

从图 3-26 中可以看出，随着 n_1 的增加，误判率也随之下降，但是其对误判率的影响也越来越小，当 n_1 为 20 时，获得的误判率小于 0.7%，所以当 n_1 取为 20 时，基于多主机特征的 P2P 流检测就能获得较低的误判率。

2. 基于通信对端类型检测的误判率

对于通信对端类型特征，需要确定式(3-2)的阈值 n_4，由于在判断对方类型时判断条件的严格性(如端口号＞10000 等)，因此 n_4 只要取很小的值就能获得较低的误判率，在图 3-27 中给出了基于对端类型特征的 P2P 检测对传统网络服务的误判率。

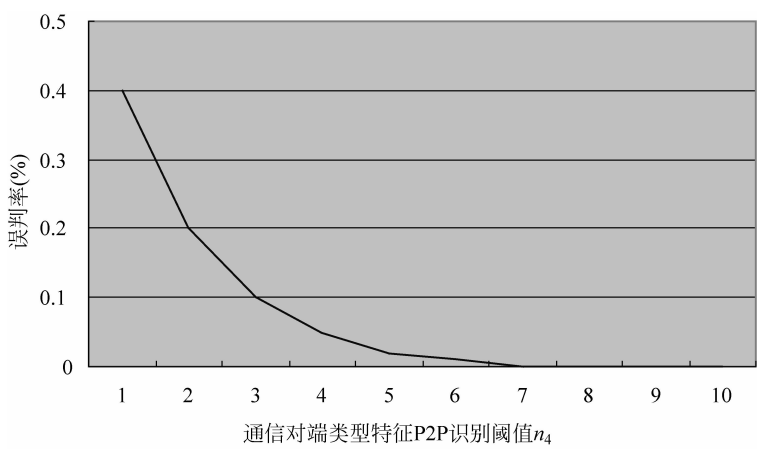


图 3-27　基于通信对端特征的 P2P 检测误判率

从图 3-27 中可以看出，随着 n_4 的增加，误判率也随着降低，当 n_4 取得 1 时，其误判率不超过 0.4%，我们认为这已经能够满足误判率的要求，所以把 n_4 的阈值设置为 1 作为测试 P2P 识别准确度的阈值。

如前所述，P2P 识别检测系统首先需要保证较低误判率，在这基础上才需要进一步提高识别的准确率，因为只要对系统中的大部分 P2P 流能够识别出并且进行控制就能够取得较好的效果，根据本节的分析以及测试结果，对 P2P 流量识别模型中的式(3-1)和式(3-2)的阈值取值分别为 $n_2=n_1\times50\%$、$n_3=n_1\times30\%$、$n_1=20$、$n_4=1$，这组阈值保证了系统的误判率小于 0.7%，达到了较低误判率的要求。

3.5.3 准确率测试分析

识别准确率主要测试系统对于 P2P 流的识别能力，既要将 P2P-CNTIM 原型系统和传统的基于 Payload 特征的识别系统之间的识别准确率进行比较，同时也要对 P2P-CNTIM 原型系统中识别 P2P 流的两种不同方法进行比较。

首先需要分析基于通信对端类型特征的 P2P 流量识别能力[24]，由于判断对方类型的严格性，因此只要取得 $n_4=1$ 就能够获得较低的误判率，根据通信对端类型进行 P2P 识别的准确率依赖于能够准确判断对方类型的检测结果以及 P2P 应用中 NAT 后内网主机占整个对端主机中的比例，为了获得这样的比例参数，在不同的主机上运行不同应用的 P2P 软件，并对其通信过程进行跟踪，经过反复的实验和统计分析，获得约有 42%的网络设备(主机)能够对测试对端类型的报文返回响应，在能够响应的网络设备中约有 36%的对端可以确定为 NAT 后的内网主机，对于 P2P 应用，根据该实验值可以计算得到判断被观察主机为 P2P 流，需要测试的平均通信对端主机数为 $1/(42\%\times36\%)\approx6.6$ 设备，也就是说如果 $n_4=1$，那么需要平均探测 7 台对端主机类型就可以判断出其为 P2P 流，由于在误判率实验过程中，系统取 $n_1=20$ 以保证较小的误判率，那么从总体上来说，通过对端类型特征来识别 P2P 流只需要相对较少的对端主机数就可以进行判别。

在实验室不同的主机上分别运行 BitTorrent、eMule、PPLive、QQLive、PPStream、Kugoo 这些常见的 P2P 软件，在这些软件中，有一部分是已知其 Payload 特征的，而另外有个别软件系统并不知道其载荷特征，实验室的其他 10 台主机使用正常的 Web、FTP 等传统网络服务，实验室中有两台主机分别运行 P2P-CNTIM 原型系统和基于 Payload 特征进行 P2P 识别的系统，两台检测 P2P 的主机的网卡均设置为混杂模式以获得整个局域网内所有的通信报文，这样就能保证两个 P2P 识别系统处理的数据包是一致的。由于部分 Payload 特征只在应用开始的时候才会出现，因此在检测开始前，需要在检测主机上先启动 P2P 识别系统。进行测试的目的是检测 P2P 识别的准确性，所以设定检测的时间为 10 个单位分钟，在检测时间结束时，统计识别出的 P2P 流的主机数量和产生 P2P 流的总的主机数量的比例，重复这样的实验并计算平均值，用来作为系统的 P2P 识别准确率。为了比较通信对端类型特征和多主机特征对识别 P2P 流的效果，在实验中分别记录了基于对端类型特征检测出的结果和基于多主机特征检测出的结果以及其检测出的时间。实验的结果如表 3-4 所示。

表 3-4　基于 Payload 特征 P2P 识别系统和原型系统识别准确率对比

序号	应用名称	传统 Payload 识别准确率	基于 P2P-CNTIM 的 P2P 识别准确率		
			对端类型特征	多主机特征	总计
1	BitTorrent	91.25%	46.53%	37.26%	83.79%
2	eMule	90.32%	50.27%	36.89%	87.16%
3	PPLive	88.94%	42.35%	40.22%	82.57%
4	QQLive	93.88%	35.88%	52.93%	88.81%
5	PPStream	90.53%	41.73%	44.58%	86.31%
6	未知 P2P	0	35.27%	48.93%	84.20%

从表 3-4 的数据可以看出，对于已知类型的 P2P 应用，传统 Payload 特征的识别效率比较高，一般其识别准确率高于 P2P-CNTIM 原型系统，这主要是因为基于 Payload 特征检测对所有经过的数据包都进行匹配检测，当 Payload 特征匹配时，就将其标识为 P2P 应用，这对于特征明显的 P2P 应用具有较高的检测准确率，而 P2P-CNTIM 原型系统的数据包存在时间有效性，只对部分的数据包进行检测，且易受到复杂的网络情况以及用户的使用情况的影响，所以有可能导致 P2P 识别准确性有较小的降低。

然而对未知的 P2P 流，Payload 检测方法无法检测出该 P2P 流，但是基于 P2P-CNTIM 的 P2P 识别系统仍然能正确将其划分为 P2P 流，经过进一步分析后我们获得了该未知 P2P 流是某台主机上用户运行 Kugoo 软件产生的，对于识别未知 P2P 流，基于 P2P-CNTIM 的 P2P 识别系统具有明显的优点。

在表 3-4 中，P2P-CNTIM 原型系统通过使用通信对端类型检测技术，能够识别出 30%～55%的 P2P 流，充分说明了基于通信对端类型检测技术的有效性，同时该技术通过和多主机特征进行互补，使得原型系统对 P2P 流量识别具有较高的准确率。根据分别记录的基于多主机特征的检测结果以及基于对端类型特征检测出的结果，得到了多主机特征和通信对端类型检测各自的 P2P 流量识别准确率以及总的识别准确率，如图 3-28 所示。

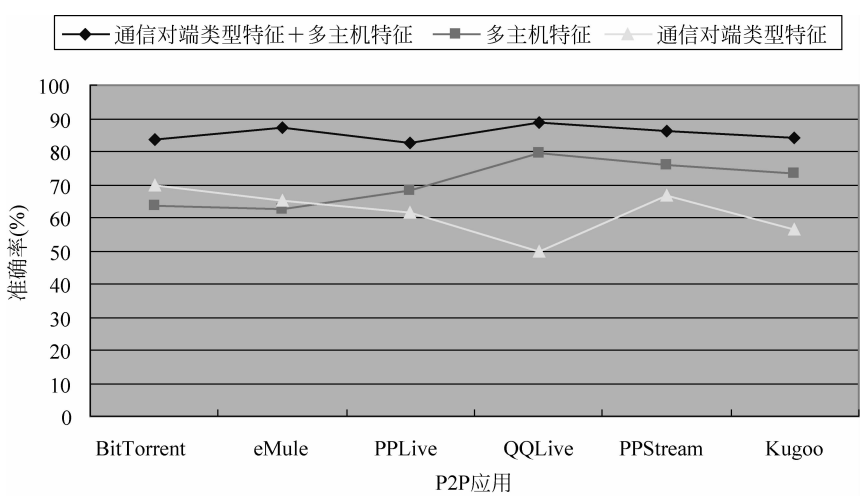


图 3-28　多主机特征、通信对端类型特征、双重特征的识别准确率

有两个因素决定了基于流量特征的 P2P 流量识别技术存在准确性不够高的缺点。第一个因素是 P2P 流量特征不一定唯一。很多流量特征都不是 P2P 流量唯一的，其他应用也有可能表现出这种流量特征来。因此，该方法存在误判问题，即将不是 P2P 流量的网络流，误认为是 P2P 流量。第二个因素是网络环境复杂。例如，由于不对称路由和丢包、重传现象的存在，导致无法精确确定流量特征，从而有可能对 P2P 流量检测的精确度造成影响。图 3-28 实验数据表明 P2P-CNTIM 原型系统由于同时采用了多主机特征以及通信对端类型特征来进行 P2P 流量识别，比使用单独的特征检测具有更高的准确性。这主要是因为对于多主机特征来说，如果 P2P 的应用并不占用大量的端口，或者用户对连接数进行了限制，那么就容易出现漏判的情况[25]，而对于基于通信对端类型进行 P2P 流量识别的方法来说，由于判断对方类型的严格性，有可能有的 P2P 应用只使用了低端口来进行通信，因此也会造成漏判的情况。然而通过将这两个特征有机地结合起来，共同实现对 P2P 流的检测，则可以获得较高的准确率。

由于 P2P 识别、控制管理系统需要有较低的误判率和漏检率，在保证误判率的基础上，只要识别出系统中的大部分 P2P 流就可以通过对其进行流量控制而达到较好的网络管理效果，实验数据表明 P2P-CNTIM 原型系统具有较高的识别准确率，且其具有对未知类型或者加密协议的 P2P 流检测的能力，而基于 Payload 特征检测的方法对未知类型或者加密协议的 P2P 流检测将完全失效，另外，基于 Payload 特征检测的方法由于很多特征都对检测包的顺序及位置有严格的要求，因此它需要先于 P2P 应用运行才能取得较好的效果，而基于流的特征来对 P2P 进行识别则不受到此限制[26]。

3.5.4 识别效率分析

P2P 识别、控制管理系统一般部署在企业的局域网出口或者 ISP 的汇接网络设备附近，由于其要监控局域网内所有的主机，因此处理的网络报文数据量较大，对于 P2P 识别系统来说，识别效率将成为衡量 P2P 识别系统的关键指标，基于 Payload 特征检测方法的执行效率主要取决于匹配操作执行的效率，为了量化比较算法执行的效率，我们提出一个抽象的衡量操作复杂度的单位，称为操作基本单元，每个特征串匹配可以看成由若干操作基本单元组成[27]。假设在一段时间内，处理的未知报文总数为 N，其中 P2P 报文所占的比例为 p，系统中的所有 Payload 特征串包含的操作基本单元为 $k_{\max}$，则 N 个报文执行匹配操作可以用式(3-5)近似表示为：

$$O_{\text{Payload}} = \sum_{i=1}^{N*p} K_i + \sum_{i=1}^{N*(1-p)} k_{\max} \tag{3-5}$$

式(3-5)中，K_i 表示第 i 个符合某个特征属性的报文对应的匹配操作包含的操作基本单元数。公式有两部分组成，第一部分表示所有 P2P 报文经过 Payload 特征串的匹配包含的操作基本单元总数，第二部分表示 N 个报文中非 P2P 报文进行匹配包含的操作基本单元总数。

假设在 P2P-CNTIM 原型系统中通过通信对端类型特征检测出来的 P2P 流所占的百分比为 P_n，则对于 N 个报文执行的操作可近似表示为：

$$O_{\text{P2P-CNTRM}}=\sum_{i=1}^{N*p*P_n}L_i+\sum_{i=1}^{N*p*(1-P_n)}M_i+\sum_{i=1}^{N*p}K_i+\sum_{i=1}^{N*(1-p)}(L_i+M_i)$$

$$\leqslant N*(L_i+M_i)+\sum_{i=1}^{N*p}K_i \tag{3-6}$$

式(3-6)中，L_i 表示对报文 i 使用通信对端类型特征 P2P 识别算法包含的操作基本单元数；M_i 表示对报文 i 使用多主机特征 P2P 识别算法包含的操作基本单元数。对于 L_i 来说，其包含了在通信对端类型检测子模块维护的 nat_test_statu 列表中进行该数据包对应的对端类型信息的查找操作以及检测一个通信对端类型需要的操作，由于整个处理的报文队列只保存单位分钟内的数据包，因此 L_i 存在一个上限；同样对于 M_i 来说，其需要统计的报文列表中的报文数也是一个单位分钟内的报文，所以 M_i 也存在一个上限。

式(3-5)和式(3-6)主要区别体现在非 P2P 流的处理上，为了对比基于 Payload 特征的 P2P 识别方法和基于通信网络拓扑特征的 P2P 识别方法执行效率，假设式(3-5)和式(3-6)相等，则可以得到式(3-7)：

$$N*(L_i+M_i)=N*(1-p)*k_{\max}\Rightarrow L_i+M_i=(1-p)*k_{\max}$$

$$\Rightarrow k_{\max}=(L_i+M_i)/(1-p) \tag{3-7}$$

由于 L_i+M_i 是存在上限的，也就是可以认为它们是两个有限的数值，为了使得两种识别方法具有相同的处理效率，必须使得式(3-7)能够成立，在式(3-7)中，当 P2P 的数据包比例 p 接近于 1 时，那么 $k_{\max}$ 将可以取得很大值，在这个时候，基于 Payload 特征进行 P2P 识别方法执行效率优于基于 P2P-CNTIM 的 P2P 识别方法，但是随着报文中非 P2P 比例的上升 $k_{\max}$ 将变得越来越小，所以随着非 P2P 报文比例的上升，基于 P2P-CNTIM 的 P2P 识别方法在执行效率上将逐渐优于基于 Payload 特征进行 P2P 识别方法。由于 $k_{\max}$ 是系统中所有 Payload 特征串包含的操作基本单元，所以其一般远大于 L_i+M_i，且随着 P2P 流量控制机制的引入，网络上 P2P 流量所占的比例将逐步减少。从而导致在执行效率上基于 P2P-CNTIM 的 P2P 识别方法将优于传统的基于 Payload 特征的识别方法。在实验中，我们在不同的实验主机上分别运行 P2P 应用和非 P2P 应用，并不断提高非 P2P 流所占的比例，每次捕获处理完 10000 个数据包就把 Payload 匹配执行次数、通信对端类型特征检测次数，以及多主机特性检测次数分别记录下来用来作为整个 P2P 识别系统的操作基本单元，对多主机特征统计操作和通信对端类型特征测试操作看作和 Payload 特征字匹配相同的操作单元，根据记录的结果绘制成图 3-29。

在图 3-29 中，用匹配次数总数来近似比较基于 P2P-CNTIM 的 P2P 识别方法和基于 Payload 特征的识别方法执行效率，实验的数据表明对于相同数量的数据包处理，随着非 P2P 数据所占的比例的上升，基于 Payload 特征的识别方法匹配比较次数增长迅速[28]。对于基于 P2P-CNTIM 的 P2P 识别方法来说，由于其先要进行对端类型特征匹配和多主机特征匹配，因此在 P2P 流占用的比例比较高时，与基于 Payload 特征的识别方法相比，需要进行对端类型特征匹配和多主机特征匹配两个额外的操作步骤，所以匹配的次数比基于 Payload 特征的识别方法要高，但是随着 P2P 流占的比例的减少，其执行 Payload 操作的概率也就越来越少，系统主要是进行通信对端类型特征匹配和多主机特征匹配操作，所以其总的操作数将逐渐降低[29]。需要说明的是，尽管该实验过程有很多的限制条件，与真实情况会有一些差异，但确实体现了这两种方法在执行效率上本质的区别。

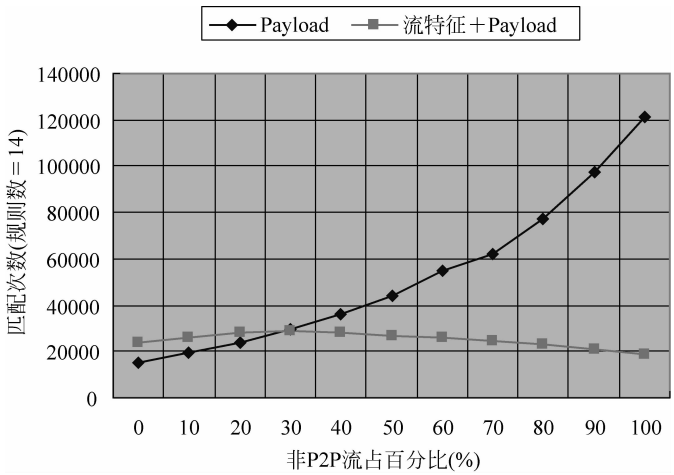


图 3-29 原型系统和基于 Payload 特征的 P2P 识别效率对比

3.6 本章小结

本章分析了 P2P 在 Internet 上的通信网络拓扑特征，对目前已经提出的 P2P 识别技术进行了总结，并分析了各种技术的使用限制。

针对 P2P 流的识别问题，从 P2P 独有的通信网络拓扑结构出发，根据 P2P 应用的多主机特征以及通信对端类型特征，提出了基于通信网络拓扑结构的 P2P 流量识别模型，对该模型选择的确定性特征、识别通信对端类型的核心技术、P2P 流量识别 P2P-CNTIM 的判断函数、调度机制以及核心过程进行了详细介绍。

针对 ISP 以及企业对 P2P 识别、控制管理的需求，结合基于流量识别模型 P2P-CNTIM，给出了一个 P2P 识别、控制管理系统的设计与实现。

本章最后通过实验测试，证明基于 P2P-CNTIM 的 P2P 识别、控制管理原型系统有着较低的误判率和较高的识别准确率，并且和传统的基于 Payload 特征的 P2P 识别方法相比，其在执行效率上有着明显地提高。

本章参考文献

[1] Constantinou F, Mavrommatis PBI. Identifying Known and Unknown Peer-to-Peer Traffic[J]. In: Bilof R, ed. Proc. of the 5th IEEE Int'l Symp. on Network Computing and Applications. New York: IEEE Computer Society, 2006: 93-102.

[2] Sun Zhixin, Yan Xiaojuan, Chen Songlesun. Research on P2P Flow Identification Model Based on Communicating Network Topology[J]. Scientific Research and Essays Vol. 6(18). 3822-3833, 1 September, 2011.

[3] D Stutzbach, R Rejaie, S Sen. Characterizing Unstructured Overlay Topologies in Modern P2P File-Sharing Systems[J]. Networking, IEEE/ACM Transactions, 2008(Volume 16): 267-280.

[4] Zhao R. The Research and Implementation of P2P Traffic Identification Based on Feature String[MS.

Thesis]. Chengdu: University of Electronic Science and Technology of China, 2009.

[5] T Wu, Y Xi, C Li, et al. Study on the Fast Response and Adaptability Peer-to-Peer Network Based on Optimized Meridian and Gnutella[J]. in: Computational Intelligence and Software Engineering, 2009: 1-4.

[6] Skype: Free Internet Telephony that Just Works[EB/OL]. http://www.skype.com.

[7] A. Rowstron, P. Druschel. Pastry: Scalable, Decentraized Object Location and Routing for Large-scale Peer-to-Peer Systems[J]. Lecture Notes in Computer Science, 2001: 329-350.

[8] Stoica I, Morris R, Karger D, et al. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications[C]. //ACM SIGCOMM Computer Communication Review. ACM, 2001, 31(4): 149-160.

[9] Ratnasamy S, Francis P, Handley M, et al. A Scalable Content Addressable Network[M]. ACM, 2001: 161-172.

[10] B. Y. Zhao, J. D. Kubiatowicz, A. D. Joseph. Tapestry. An Infrastructure for Fault-tolerant Wide-area Location and Routing[R]. Technical Report UCB/CSD-01-1141, Computer Science Division, U. C. Berkeley, 2001.

[11] Comtantinou F, Mavromrnatis P. Identifying Known and Unknown Peer-to-Peer Trafic[C]. Network Computing and Applications. 2006. NCA 2006. Fifth IEEE International Symposium onVolume, 2006: 93-102.

[12] Bolla R, Canini M, Rapuzzi R. On the Double-Faced Nature of P2P Traffic Department of Communication[C]. In Proceedings of the Sixteenth Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2008: 524-530.

[13] CacheLogic. CacheLogic-Advanced Solutions for P2P Networks[EB/OL]. http://www.cachelogic.com, 2006.

[14] Satoshi Ohzahata1m, Yoichi Hagiwara1, Matsuaki Terada1, et al. A Traffic Identification Method and Evaluations for a Pure P2P Application Source[J]. Lecture Notes in Computer Science. 2005, 3431: 55-68.

[15] Holger Bleul, Erwin P. Rathgeb, Stefan Zilling. Evaluation of an Efficient Measurement Concept for P2P Multiprotocol Traffic Analysis[A]. In: Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications[C]. Cavtat: IEEE Computer Society Press, 2006: 414-423.

[16] Holger Bleul, Erwin P. Rathgeb, Stefan Zilling. Advanced P2P Multiprotocol Traffic Analysis Based on Application Level Signature Detection [A]. In: Proceedings of the 12th International Telecommunications Network Strategy and Planning Symposium[C]. New Dehli: IEEE Press, 2006: 1-6.

[17] 宫婧,孙知信,顾强. 基于行为特征描述的 P2P 流量识别方法研究[J]. 小型微型计算机系统,2007, 28(1): 48-53.

[18] 孙知信,宫婧,曾骁,等. 一种 P2P 数据流的识别方法、装置和系统. 中国, 200810189131[P]. 2008-12-29.

[19] 杨锐. 特征字符串匹配在 P2P 流量控制中的应用[J]. 科技信息,2007,2(10): 158-159.

[20] Chen QZ, Shao B, Chen C. Design and Implementation of P2P Traffic Identification System Based on Compound Characteristics[J]. Journal of Southest University (Natural Science Edition), 2008, 38(S1): 109-113.

[21] Lijuan Zhou, Zhitong Li, Bin Liu. P2P Traffic Identification by TCP Flow Anylasis[A]. In: Proceedings of IWNA'06[C]. Shenyang: IEEE Press, 2006: 1-2.

[22] Matsuda. T, Nakamura. F, Wakahara. Y, et al. Traffic Features Fit for P2P Discrimination[J].

IEEE,Information and Telecommunication Technologies. 2005，9(10)：230-235.

[23] Dedinski，I. DeMeer，H. Han,et al. Cross-Layer Peer-to-Peer Traffic Identification and Optimization Based on Active Networking[A]. In：Proceedings of the Seventh Annual International Working Conference on Active and Programmable Networks(IWAN'05)[C]，CICA，Sophia Antipolis，French Riviera，La Cote d'Azur，France，2005，21-23.

[24] Moore A，Zuev D. Internet Traffic Classification Using Bayesian Analysis Techniques[A]. In：Proceedings of SIGMETR ICS'05[C]. New York：ACM Press，2005：50-60.

[25] Erman J，Arl Ittm，An Irban M. Traffic Classification Using Clustering Algorithms[A]. In：Proceedings of SIGCOMM Workshop on Mining Network Data[C]. New York：ACM Press，2006：11-15.

[26] 孙知信,宫婧. 一种基于流特性描述的 P2P 流量模糊识别方法[J]. 计算机学报,2008，31(07)：1252-1260.

[27] Gonzalez-Castano，F. J. Rodriguez-Hernandez，P. S. Martinez-Alvarez，et al. Support Vector Machine Detection of P2P Traffic[A]. IEEE Computational Intelligence for Measurement Systems and Applications，IEEE International Conference on July 2006[C]，2006：103-108.

[28] Rui Wang，Ang Liu，Yuexiang Yang,et al. Solving the App-Level Classification Problem of P2P Traffic Via Optimized Support Vector Machines[A]. In：Proceedings of ISDA'06[C]. Jinan：IEEE Press，2006：534-539.

[29] 刘竟,郑志彬,刘廷永,等. 网络数据流量识别系统及方法. 中国，200510101365[P]. 2005-1-119.

[30] 陈松乐. 基于通信网络拓扑结构的 P2P 流量识别模型研究[D]. 南京：南京邮电大学,2009.